A Survey of Research and Practices of Network-on-Chip

TOBIAS BJERREGAARD AND SHANKAR MAHADEVAN

Technical University of Denmark

The scaling of microchip technologies has enabled large scale systems-on-chip (SoC). Network-on-chip (NoC) research addresses global communication in SoC, involving (i) a move from computation-centric to communication-centric design and (ii) the implementation of scalable communication structures. This survey presents a perspective on existing NoC research. We define the following abstractions: system, network adapter, network, and link to explain and structure the fundamental concepts. First, research relating to the actual network design is reviewed. Then system level design and modeling are discussed. We also evaluate performance analysis techniques. The research shows that NoC constitutes a unification of current trends of intrachip communication rather than an explicit new alternative.

Categories and Subject Descriptors: A.1 [**Introductory and Survey**]; B.4.3 [**Input/Output and Data-Communications**]: Interconnections; B.7.1 [**Integrated Circuits**]: Types and Design Styles; C.5.4 [**Computer System Implementation**]: VLSI Systems; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design; C.0 [**General**]: *System Architectures*

General Terms: Design

Additional Key Words and Phrases: Chip-area networks, communication-centric design, communication abstractions, GALS, GSI design, interconnects, network-on-chip, NoC, OCP, on-chip communication, SoC, sockets, system-on-chip, ULSI design

1. INTRODUCTION

Chip design has four distinct aspects: computation, memory, communication, and I/O. As processing power has increased and data intensive applications have emerged, the challenge of the communication aspect in single-chip systems, Systems-on-Chip (SoC), has attracted increasing attention. This survey treats a prominent concept for communication in SoC known as Network-on-Chip (NoC). As will become clear in the following, NoC does not constitute an explicit new alternative for intrachip communication but is rather a concept which presents a unification of on-chip communication solutions.

In this section, we will first briefly review the history of microchip technology that has led to a call for NoC-based designs. With our minds on intrachip communication,

---

This paper is a joint author effort, authors in alphabetical order.

S. Mahadevan was funded by SoC-MOBINET (IST-2000-30094), Nokia and the Thomas B. Thrige Foundation.

Authors' address: Technical University of Denmark, Informatics and Mathematical Modelling, Richard Petersens Plads, Building 321, DK-2800 Lyngby, Denmark; email:{tob,sm}@imm.dtu.dk.

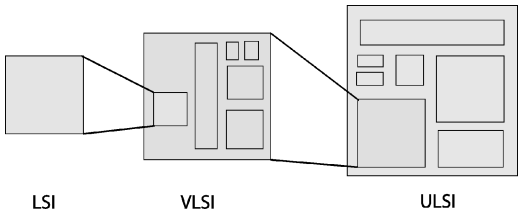

Fig. 1. When a technology matures, it leads to a paradigm shift in system scope. Shown here is the chip scope in LSI, VLSI, and ULSI, the sequence of technologies leading to the enabling of SoC designs.

we will then look at a number of key issues of large-scale chip design and finally show how the NoC concept provides a viable solution space to the problems presently faced by chip designers.

1.1. IntraSoC Communication

The scaling of microchip technologies has lead to a doubling of available processing resources on a single chip every second year. Even though this is projected to slow down to a doubling every three years in the next few years for fixed chip sizes [ITRS 2003], the exponential trend is still in force. Though the evolution is continuous, the system level focus, or system scope, moves in steps. When a technology matures for a given implementation style, it leads to a paradigm shift. Examples of such shifts are moving from room- to rack-level systems (LSI-1970s) and later from rack- to board-level systems (VLSI-1980s). Recent technological advances allowing multimillion transistor chips (currently well beyond 100M) have led to a similar paradigm shift from board- to chip-level systems (ULSI-1990s). The scope of a single chip has changed accordingly as illustrated in Figure 1. In LSI systems, a chip was a component of a system module (e.g., a bitslice in a bitslice processor), in VLSI systems, a chip was a system-level module (e.g., a processor or a memory), and in ULSI systems, a chip constitutes an entire system (hence the term System-on-Chip). SoC opens up the feasibility of a wide range of applications making use of massive parallel processing and tightly interdependent processes, some adhering to real-time requirements, bringing into focus new complex aspects of the underlying communication structure. Many of these aspects are addressed by NoC.

There are multiple ways to approach an understanding of NoC. Readers well versed in macronetwork theory may approach the concept by adapting proven techniques from multicomputer networks. Much work done in this area during the 80s and 90s can readily be built upon. Layered communication abstraction models and decoupling of computation and communication are relevant issues. There are, however, a number of basic differences between on- and off-chip communication. These generally reflect the difference in the cost ratio between wiring and processing resources.

Historically, computation has been expensive and communication cheap. With scaling microchip technologies, this changed. Computation is becoming ever cheaper, while communication encounters fundamental physical limitations such as time-of-flight of electrical signals, power use in driving long wires/cables, etc. In comparison with off-chip, on-chip communication is significantly cheaper. There is room for lots of wires on a chip. Thus the shift to single-chip systems has relaxed system communication problems. However on-chip wires do not scale in the same manner as transistors do, and, as we shall see in the following, the cost gap between computation and communication is

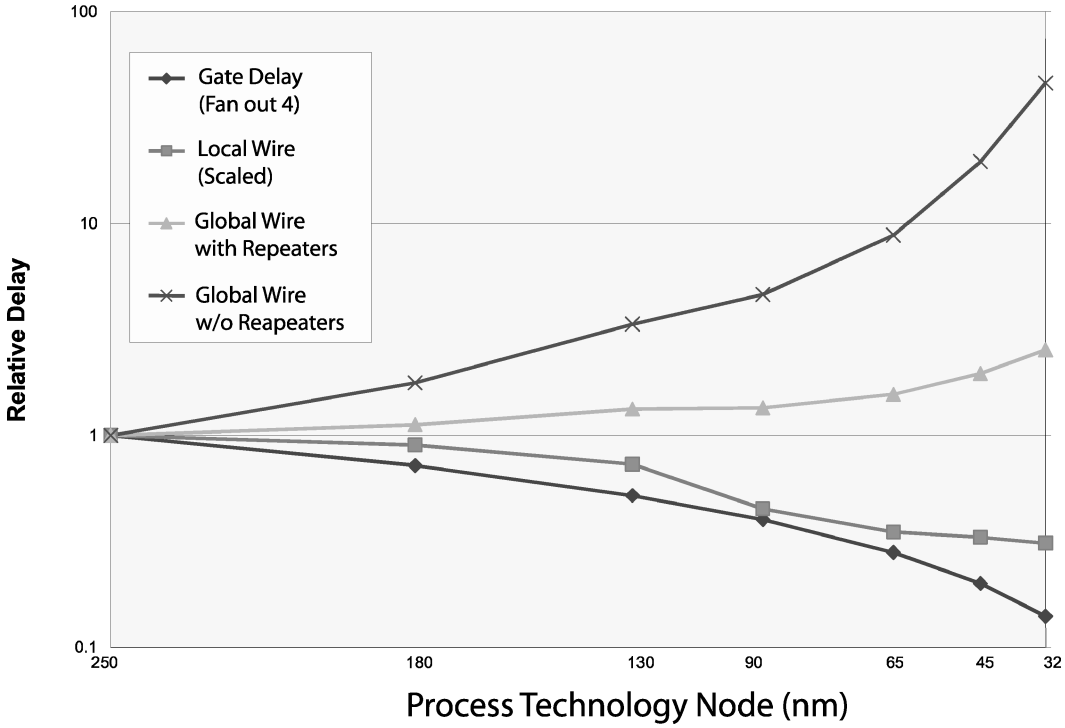

Fig. 2. Projected relative delay for local and global wires and for logic gates in technologies of the near future. [ITRS 2001].

widening. Meanwhile the differences between on- and off-chip wires make the direct scaling down of traditional multicomputer networks suboptimal for on-chip use.

In this survey, we attempt to incorporate the whole range of design abstractions while relating to the current trends of intrachip communication. With the Giga Transistor Chip era close at hand, the solution space of intrachip communication is far from trivial. We have summarized a number of relevant key issues. Though not new, we find it worthwhile to go through them as the NoC concept presents a possible unification of solutions for these. In Section 3 and 4, we will look into the details of research being done in relation to these issues, and their relevance for NoC.

—*Electrical wires*. Even though on-chip wires are cheap in comparison with off-chip wires, on-chip communication is becoming still more costly in terms of both power and speed. As fabrication technologies scale down, wire resistance per-mm is increasing while wire capacitance does not change much; the major part of the wire capacitance is due to edge capacitance [Ho et al. 2001]. For CMOS, the approximate point at which wire delays begin to dominate gate delays was the 0.25 μm generation for aluminum, and 0.18 μm for copper interconnects as first projected in SIA [1997]. Shrinking metal pitches, in order to maintain sufficient routing densities, is appropriate at the local level where wire lengths also decrease with scaling. But global wire lengths do not decrease, and, as local processing cycle times decrease, the time spent on global communication relative to the time spent on local processing increases drastically. Thus in future deep submicron (DSM) designs, the interconnect effect will definitely dominate performance [Sylvester and Keutzer 2000]. Figure 2, taken from the International Technology Roadmap for Semiconductors [ITRS 2001], shows the

projected relative delay for local wires, global wires, and logic gates in the near future. Another issue of pressing importance concerns signal integrity. In DSM technologies, the wire models are unreliable due to issues like fabrication uncertainties, crosstalk, noise sensitivity etc. These issues are especially applicable to long wires.

Due to these effects of scaling, it has become necessary to differentiate between local and global communication, and, as transistors shrink, the gap is increasing. The need for global communication schemes supporting single-chip systems has emerged.

—*System synchronization.* As chip technologies scale and chip speeds increase, it is becoming harder to achieve global synchronization. The drawbacks of the predominant design style of digital integrated circuits, that is, strict global synchrony, are growing relative to the advantages. The clocktree needed to implement a globally synchronized clock is demanding increasing portions of the power and area budget, and, even so, the clock skew is claiming an ever larger relative part of the total cycle time available [Oklobdzija and Sparsø 2002; Oberg 2003]. This has triggered work on skew-tolerant circuit design [Nedovic et al. 2003], which deals with clockskew by relaxing the need for timing margins, and on the use of optical waveguides for on-chip clock distribution [Piguet et al. 2004], for the main purpose of minimizing power usage. Still, power hungry skew adjustment techniques such as phase locked loops (PLL) and delay locked loops (DLL), traditionally used for chip-to-chip synchronization, are finding their way into single-chip systems [Kurd et al. 2001; Xanthopoulos et al. 2001].

As a reaction to the inherent limitations of global synchrony, alternative concepts such as GALS (Globally Asynchronous Locally Synchronous systems) are being introduced. A GALS chip is made up of locally synchronous islands which communicate asynchronously [Chapiro 1984; Meincke et al. 1999; Muttersbach et al. 2000]. There are two main advantageous aspects of this method. One is the reducing of the synchronization problem to a number of smaller subproblems. The other relates to the integration of different IP (Intellectual Property) cores, easing the building of larger systems from individual blocks with different timing characteristics.

—*Design productivity.* The exploding amount of processing resources available in chip design together with a requirement for shortened design cycles have pushed the productivity burden on to chip designers. Between 1997 and 2002, the market demand reduced the typical design cycle by 50%. As a result of increased chip sizes, shrinking geometries, and the availability of more metal layers, the design complexity increased 50 times in the same period [OCPIP 2003a]. To keep up with these requirements, IP reuse is pertinent. A new paradigm for design methodology is needed which allows the design effort to scale linearly with system complexity.

Abstraction at the register transfer level (RTL) was introduced with the ASIC design flow during the 90s, allowing synthesized standard cell design. This made it possible to design large chips within short design cycles, and synthesized RTL design is, at present, the defacto standard for making large chips quickly. But the availability of on-chip resources is outgrowing the productivity potential of even the ASIC design style. In order to utilize the exponential growth in number of transistors on each chip, even higher levels of abstraction must be applied. This can be done by introducing higher level communication abstractions, making a layered design methodology that enables a partitioning of the design effort into minimally interdependent subtasks. Support for this at the hardware level includes standard communication sockets which allow IP cores from different vendors to be plugged effortlessly together. This is particularly pertinent in complex multiprocessor system-on-chip (MPSoC) designs. Also, the development of design techniques to further increase the productivity of designers, is important. Electronic system level (ESL) design tools are necessary for supporting a design flow which make efficient use of such communication abstraction

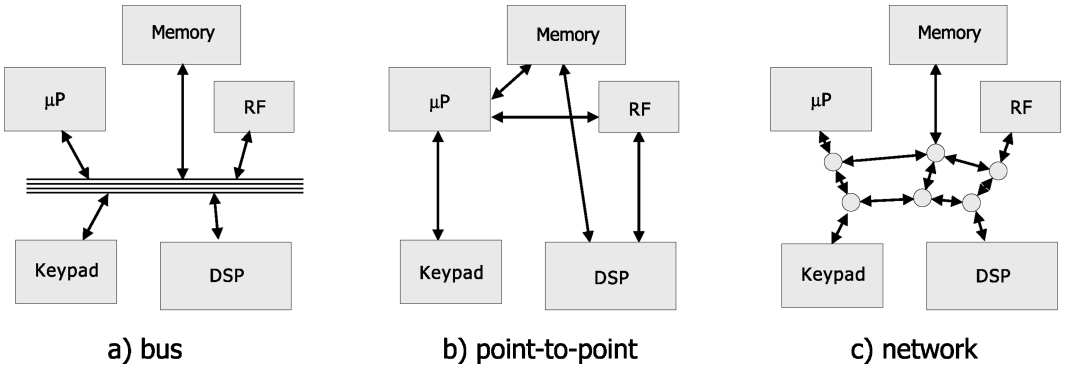

Fig. 3. Examples of communication structures in Systems-on-Chip. a) traditional bus-based communication, b) dedicated point-to-point links, c) a chip area network.

and design automation techniques and which make for seamless iterations across all abstraction levels. Pertaining to this, the complex, dynamic interdependency of data streams—arising when using a shared media for data traffic—threatens to foil the efforts of obtaining minimal interdependence between IP cores. Without special quality-of-service (QoS) support, the performance of data communication may become unwarrantly arbitrary [Goossens et al. 2005].

To ensure the effective exploitation of technology scaling, intelligent use of the available chip design resources is necessary at the physical as well as at the logical design level. The means to achieve this are through the development of effective and structured design methods and ESL tools.

As shown, the major driving factors for the development of global communication schemes are the ever increasing density of on-chip resources and the drive to utilize these resources with a minimum of effort as well as the need to counteract the physical effects of DSM technologies. The trend is towards a subdivision of processing resources into manageable pieces. This helps reduce design cycle time since the entire chip design process can be divided into minimally interdependent subproblems. This also allows the use of modular verification methodologies, that is, verification at a low abstraction level of cores (and communication network) individually and at a high abstraction level of the system as a whole. Working at a high abstraction level allows a great degree of freedom from lower level issues. It also tends towards a differentiation of local and global communication. As intercore communication is becoming the performance bottleneck in many multicore applications, the shift in design focus is from a traditional processing-centric to a communication-centric one. One top-level aspect of this involves the possibility to save on global communication resources at the application level by introducing communication aware optimization algorithms in compilers [Guo et al. 2000]. System-level effects of technology scaling are further discussed in Catthoor et al. [2004].

A standardized global communication scheme, together with standard communication sockets for IP cores, would make Lego brick-like plug-and-play design styles possible, allowing good use of the available resources and fast product design cycles.

1.2. NoC in SoC

Figure 3 shows some examples of basic communication structures in a sample SoC, for example, a mobile phone. Since the introduction of the SoC concept in the 90s, the solutions for SoC communication structures have generally been characterized by custom designed ad hoc mixes of buses and point-to-point links [Lahiri et al. 2001]. The

Table I. Bus-versus-Network Arguments (Adapted from Guerrier and Greiner [2000])

Bus Pros & Cons			Network Pros & Cons
Every unit attached adds parasitic capacitance, therefore electrical performance degrades with growth.	−	+	Only point-to-point one-way wires are used, for all network sizes, thus local performance is not degraded when scaling.
Bus timing is difficult in a deep submicron process.	−	+	Network wires can be pipelined because links are point-to-point.
Bus arbitration can become a bottleneck. The arbitration delay grows with the number of masters.	−	+	Routing decisions are distributed, if the network protocol is made non-central.
The bus arbiter is instance-specific.	−	+	The same router may be reinstantiated, for all network sizes.
Bus testability is problematic and slow.	−	+	Locally placed dedicated BIST is fast and offers good test coverage.
Bandwidth is limited and shared by all units attached.	−	+	Aggregated bandwidth scales with the network size.
Bus latency is wire-speed once arbiter has granted control.	+	−	Internal network contention may cause a latency.
Any bus is almost directly compatible with most available IPs, including software running on CPUs.	+	−	Bus-oriented IPs need smart wrappers. Software needs clean synchronization in multiprocessor systems.
The concepts are simple and well understood.	+	−	System designers need reeducation for new concepts.

bus builds on well understood concepts and is easy to model. In a highly interconnected multicore system, however, it can quickly become a communication bottleneck. As more units are added to it, the power usage per communication event grows as well due to more attached units leading to higher capacitive load. For multimaster busses, the problem of arbitration is also not trivial. Table I summarizes the pros and cons of buses and networks. A crossbar overcomes some of the limitations of the buses. However, it is not ultimately scalable and, as such, it is an intermediate solution. Dedicated point-to-point links are optimal in terms of bandwidth availability, latency, and power usage as they are designed especially for this given purpose. Also, they are simple to design and verify and easy to model. But the number of links needed increases exponentially as the number of cores increases. Thus an area and possibly a routing problem develops.

From the point of view of design-effort, one may argue that, in small systems of less than 20 cores, an ad hoc communication structure is viable. But, as the systems grow and the design cycle time requirements decrease, the need for more generalized solutions becomes pressing. For maximum flexibility and scalability, it is generally accepted that a move towards a shared, segmented global communication structure is needed. This notion translates into a data-routing network consisting of communication links and routing nodes that are implemented on the chip. In contrast to traditional SoC communication methods outlined previously, such a distributed communication media scales well with chip size and complexity. Additional advantages include increased aggregated performance by exploiting parallel operation.

From a technological perspective, a similar solution is reached: in DSM chips, long wires must be segmented in order to avoid signal degradation, and busses are implemented as multiplexed structures in order to reduce power and increase responsiveness. Hierarchical bus structures are also common as a means to adhere to the given communication requirements. The next natural step is to increase throughput by pipelining

these structures. Wires become pipelines and bus-bridges become routing nodes. Expanding on a structure using these elements, one gets a simple network.

A common concept for segmented SoC communication structures is based on networks. This is what is known as Network-on-Chip (NoC) [Agarwal 1999; Guerrier and Greiner 2000; Dally and Towles 2001; Benini and Micheli 2002; Jantsch and Tenhunen 2003]. As presented previously, the distinction between different communication solutions is fading. NoC is seen to be a unifying concept rather than an explicit new alternative. In the research community, there are two widely held perceptions of NoC: (i) that NoC is a subset of SoC, and (ii) that NoC is an extension of SoC. In the first view, NoC is defined strictly as the data-forwarding communication fabric, that is, the network and methods used in accessing the network. In the second view NoC is defined more broadly to also encompass issues dealing with the application, system architecture, and its impact on communication or vice versa.

1.3. Outline

The purpose of this survey is to clarify the NoC concept and to map the scientific efforts made into the area of NoC research. We will identify general trends and explain a range of issues which are important for state-of-the-art global chip-level communication. In doing so, we primarily take the first view of NoC, that is, that it is a subset of SoC, to focus and structure the diverse discussion. From our perspective, the view of NoC as an extension of SoC muddles the discussion with topics common to any large-scale IC design effort such as partitioning and mapping application, hardware/software codesign, compiler choice, etc.

The rest of the survey is organized as follows. In Section 2, we will discuss the basics of NoC. We will give a simple NoC example, address some relevant system-level architectural issues, and relate the basic building blocks of NoC to abstract network layers and research areas. In Section 3, we will go into more details of existing NoC research. This section is partitioned according to the research areas defined in Section 2. In Section 4, we discuss high abstraction-level issues such as design space exploration and modeling. These are issues often applicable to NoC only in the view of it as an extension of SoC, but we treat specifically issues of relevance to NoC-based designs and not to large scale IC designs in general. In Section 5, performance analysis is addressed. Section 6 presents a set of case studies describing a number of specific NoC implementations, and Section 7 summarizes the survey.

2. NOC BASICS

In this section, the basics of NoC are uncovered. First a component-based view will be presented, introducing the basic building blocks of a typical NoC. Then we will look at system-level architectural issues relevant to NoC-based SoC designs. After this, a layered abstraction-based view will be presented, looking at network abstraction models, in particular, OSI and the adaption of such for NoC. Using the foundations established in this section, we will go into further details of specific NoC research in Section 3.

2.1. A Simple NoC Example

Figure 4 shows a sample NoC structured as a 4-by-4 grid which provides global chip-level communication. Instead of busses and dedicated point-to-point links, a more general scheme is adapted, employing a grid of routing nodes spread out across the chip, connected by communication links. For now, we will adapt a simplified perspective in

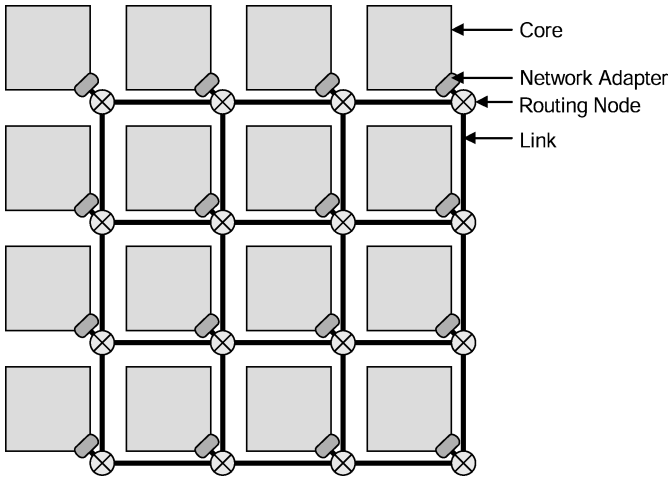

Fig. 4. Topological illustration of a 4-by-4 grid structured NoC, indicating the fundamental components.

which the NoC contains the following fundamental components.

—*Network adapters* implement the interface by which *cores* (IP blocks) connect to the NoC. Their function is to decouple computation (the cores) from communication (the network).

—*Routing nodes* route the data according to chosen protocols. They implement the routing strategy.

—*Links* connect the nodes, providing the raw bandwidth. They may consist of one or more logical or physical channels.

Figure 4 covers only the topological aspects of the NoC. The NoC in the figure could thus employ packet or circuit switching or something entirely different and be implemented using asynchronous, synchronous, or other logic. In Section 3, we will go into details of specific issues with an impact on the network performance.

2.2. Architectural Issues

The diversity of communication in the network is affected by architectural issues such as system composition and clustering. These are general properties of SoC but, since they have direct influence on the design of the system-level communication infrastructure, we find it worthwhile to go through them here.

Figure 5 illustrates how system composition can be categorized along the axes of *homogenity* and *granularity* of system cores. The figure also clarifies a basic difference between NoC and networks for more traditional parallel computers; the latter have generally been homogeneous and coarse grained, whereas NoC-based systems implement a much higher degree of variety in composition and in traffic diversity.

Clustering deals with the localization of portions of the system. Such localization may be logical or physical. Logical clustering can be a valuable programming tool. It can be supported by the implementation of hardware primitives in the network, for example, flexible addressing schemes or virtual connections. Physical clustering, based on preexisting knowledge of traffic patterns in the system, can be used to minimize global communication, thereby minimizing the total cost of communicating, power and performancewise.

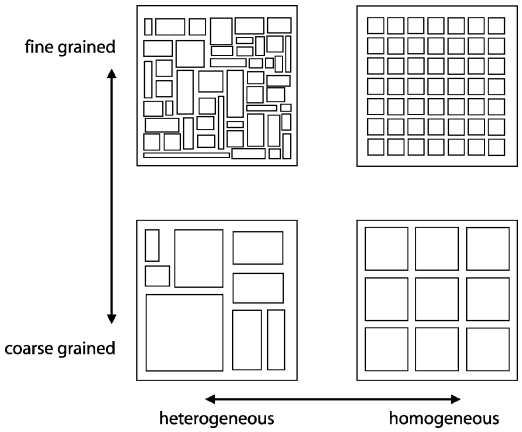

Fig. 5. System composition categorized along the axes of homogenity and granularity of system components.

Generally speaking, *reconfigurability* deals with the ability to allocate available resources for specific purposes. In relation to NoC-based systems, reconfigurability concerns how the NoC, a flexible communication structure, can be used to make the system reconfigurable from an application point of view. A configuration can be established for example, by programming connections into the NoC. This resembles the reconfigurability of an FPGA, though NoC-based reconfigurability is most often of coarser granularity. In NoC, the reconfigurable resources are the routing nodes and links rather than wires.

Much research work has been done on architecturally-oriented projects in relation to NoC-based systems. The main issue in architectural decisions is the balancing of flexibility, performance, and hardware costs of the system as a whole. As the underlying technology advances, the trade-off spectrum is continually shifted, and the viability of the NoC concept has opened up to a communication-centric solution space which is what current system-level research explores.

At one corner of the architecural space outlined in Figure 5, is the Pleiades architecture [Zhang et al. 2000] and its instantiation, the Maia processor. A microprocessor is combined with a relatively fine-grained heterogeneous collection of ALUs, memories, FPGAs, etc. An interconnection network allows arbitrary communication between modules of the system. The network is hierarchical and employs clustering in order to provide the required communication flexibility while maintaining good energy-efficiency.

At the opposite corner are a number of works, implementing homogeneous coarse-grained multiprocessors. In Smart Memories [Mai et al. 2000], a hierarchical network is used with physical clustering of four processors. The flexibility of the local cluster network is used as a means for reconfigurability, and the effectiveness of the platform is demonstrated by mimicking two machines on far ends of the architectural spectrum, the Imagine streaming processor and Hydra multiprocessor, with modest performance degradation. The global NoC is not described, however. In the RAW architecture [Taylor et al. 2002], on the other hand, the NoC which interconnects the processor tiles is described in detail. It consists of a static network, in which the communication is preprogrammed cycle-by-cycle, and a dynamic network. The reason for implementing two physically separate networks is to accommodate different types of traffic in general purpose systems (see Section 4.3 concerning traffic characterization). The Eclipse [Forsell 2002] is another similarly distributed multiprocessor architecture

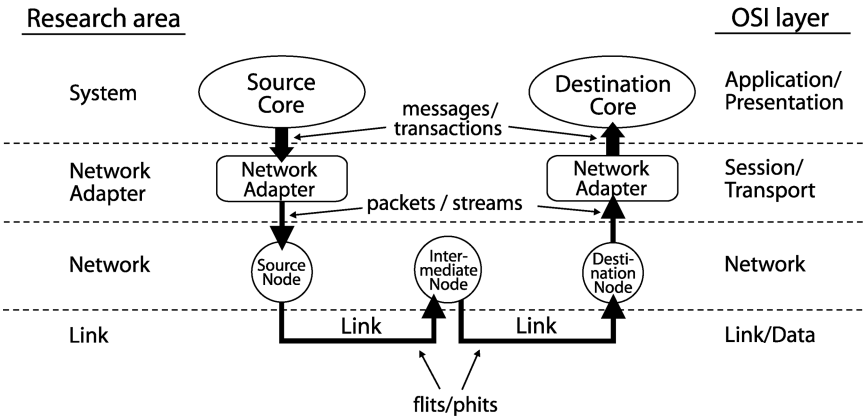

Fig. 6. The flow of data from source to sink through the NoC components with an indication of the types of datagrams and research area.

in which the interconnection network plays an important role. Here, the NoC is a key element in supporting a sofisticated parallel programming model.

2.3. Network Abstraction

The term NoC is used in research today in a very broad sense ranging from gate-level physical implementation, across system layout aspects and applications, to design methodologies and tools. A major reason for the widespread adaptation of network terminology lies in the readily available and widely accepted abstraction models for networked communication. The OSI model of layered network communication can easily be adapted for NoC usage as done in Benini and Micheli [2001] and Arteris [2005]. In the following, we will look at network abstraction, and make some definitions to be used later in the survey.

To better understand the approaches of different groups involved in NoC, we have partitioned the spectrum of NoC research into four areas: 1) system, 2) network adapter, 3) network and 4) link research. Figure 6 shows the flow of data through the network, indicating the relation between these research areas, the fundamental components of NoC, and the OSI layers. Also indicated is the basic datagram terminology.

The *system* encompasses applications (processes) and architecture (cores and network). At this level, most of the network implementation details may still be hidden. Much research done at this level is applicable to large scale SoC design in general. The *network adapter* (NA) decouples the cores from the network. It handles the end-to-end flow control, encapsulating the *messages* or *transactions* generated by the cores for the routing strategy of the Network. These are broken into *packets* which contain information about their destination, or connection-oriented *streams* which do not, but have had a path setup prior to transmission. The NA is the first level which is network aware. The *network* consists of the routing nodes, links, etc, defining the topology and implementing the protocol and the node-to-node flow control. The lowest level is the *link* level. At this level, the basic datagram are *flits* (flow control units), node level atomic units from which packets and streams are made up. Some researchers operate with yet another subdivision, namely *phits* (physical units), which are the minimum size datagram that can be transmitted in one link transaction. Most commonly flits and phits are equivalent, though in a network employing highly serialized links, each flit could be made up of a sequence of phits. Link-level research deals mostly with

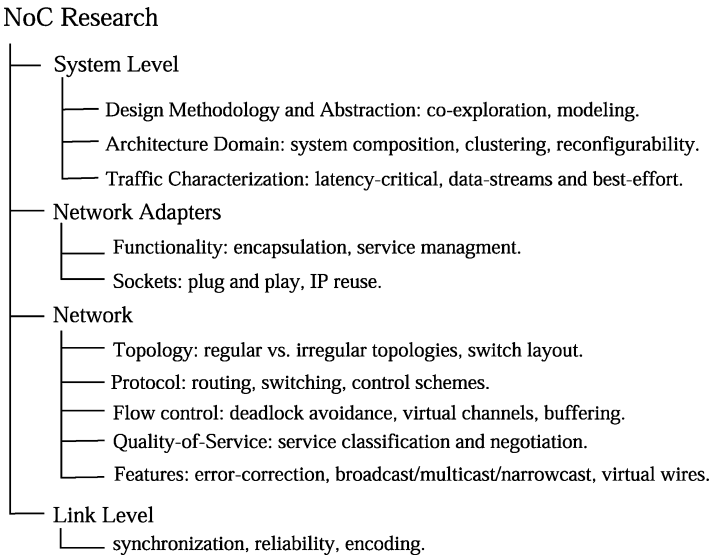

Fig. 7. NoC research area classification. This classification, which also forms the structure of Section 3, is meant as a guideline to evaluate NoC research and not as a technical categorization.

encoding and synchronization issues. The presented datagram terminology seems to be generally accepted, though no standard exists.

In a NoC, the layers are generally more closely bound than in a macronetwork. Issues arising often have a more physically-related flavor even at the higher abstraction levels. OSI specifies a protocol stack for multicomputer networks. Its aim is to shield higher levels of the network from issues of lower levels in order to allow communication between independently developed systems, for example, of different manufacturers, and to allow ongoing expansion of systems. In comparison with macronetworks, NoC benefits from the system composition being completely static. The network can be designed based on knowledge of the cores to be connected and also possibly on knowledge of the characteristics of the traffic to be handled, as demonstrated in for example, Bolotin et al. [2004] and Goossens et al. [2005]. Awareness of lower levels can be beneficial as it can lead to higher performance. The OSI layers, which are defined mainly on the basis of pure abstraction of communication protocols, thus cannot be directly translated into the research areas defined here. With this in mind, the relation established in Figure 6 is to be taken as a conceptual guideline.

3. NOC RESEARCH

In this section, we provide a review of the approaches of various research groups. Figure 7 illustrates a simplified classification of this research. The text is structured based on the layers defined in Section 2.3. Since we consider NoC as a subset of SoC, system-level research is dealt with separately in Section 4.

3.1. Network Adapter

The purpose of the network adapter (NA) is to interface the core to the network and make communication services transparently available with a minimum of effort from

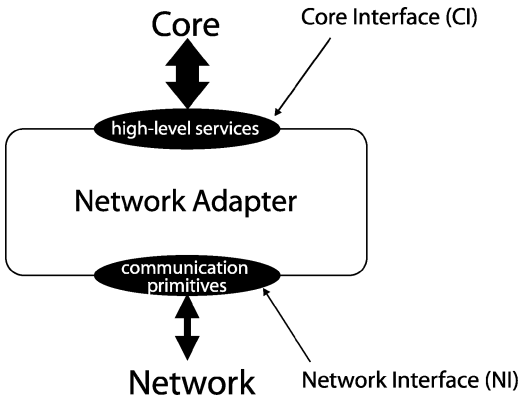

Fig. 8. The network adapter (NA) implements two interfaces, the core interface (CI) and the network interface (NI).

the core. At this point, the boundary between computation and communication is specified.

As illustrated in Figure 8, the NA component implements a core interface (CI) at the core side and a network interface (NI) at the network side. The function of the NA is to provide high-level communication services to the core by utilizing primitive services provided by the network hardware. Thus the NA *decouples* the core from the network, implementing the network end-to-end flow control, facilitating a layered system design approach. The level of decoupling may vary. A high level of decoupling allows for easy reuse of cores. This makes possible a utilization of the exploding resources available to chip designers, and greater design productivity is achieved. On the other hand, a lower level of decoupling (a more network aware core) has the potential to make more optimal use of the network resources.

In this section, we first address the use of standard sockets. We then discuss the abstract functionality of the NA. Finally, we talk about some actual NA implementations which also address issues related to timing and synchronization.

3.1.1. Sockets. The CI of the NA may be implemented to adhere to a SoC socket standard. The purpose of a socket is to orthogonalize computation and communication. Ideally a socket should be completely NoC implementation agnostic. This will facilitate the greatest degree of reusability because the core adheres to the specification of the socket alone, independently of the underlying network hardware. One commonly used socket is the *Open Core Protocol* (OCP) [OCPIP 2003b; Haverinen et al. 2002]. The OCP specification defines a flexible family of memory-mapped, core-centric protocols for use as a native core interface in on-chip systems. The three primary properties envisioned in OCP include (i) architecture independent design reuse, (ii) feature-specific socket implementation, and (iii) simplification of system verification and testing. OCP addresses not only dataflow signaling, but also uses related to errors, interrupts, flags and software flow control, control and status, and test. Another proposed standard is the *Virtual Component Interface* (VCI) [VSI Alliance 2000] used in the SPIN [Guerrier and Greiner 2000] and Proteo [Siguenza-Tortosa et al. 2004] NoCs. In Radulescu et al. [2004], support for the Advanced eXtensible Interface (AXI) [ARM 2004] and Device Transaction Level (DTL) [Philips Semiconductors 2002] protocols was also implemented in an NA design.

3.1.2. NA Services. Basically, the NA provides *encapsulation* of the traffic for the underlying communication media and *management* of services provided by the network. Encapsulation involves handling of end-to-end flow control in the network. This may include global addressing and routing tasks, reorder buffering and data acknowledgement, buffer management to prevent network congestion, for example, based on credit, packet creation in a packet-switched network, etc.

Cores will contend for network resources. These may be provided in terms of service quantification, for example, bandwidth and/or latency guarantees (see also Sections 3.2.4 and 5). Service management concerns setting up circuits in a circuit-switched network, bookkeeping tasks such as keeping track of connections, and matching responses to requests. Another task of the NA could be to negotiate the service needs between the core and the network.

3.1.3. NA Implementations. A clear understanding of the role of the NA is essential to successful NoC design. Muttersbach et al. [2000] address synchronization issues, proposing a design of an asynchronous wrapper for use in a practical GALS design. Here the synchronous modules are equipped with asynchronous wrappers which adapt their interfaces to the self-timed environment. The packetization occurs within the synchronous module. The wrappers are assembled from a concise library of predesigned technology-independent elements and provide high speed data transfer. Another mixed asynchronous/synchronous NA architecture is proposed in Bjerregaard et al. [2005]. Here, a synchronous OCP interface connects to an asynchronous, message-passing NoC. Packetization is performed in the synchronous domain, while sequencing of flits is done in the asynchronous domain. This makes the sequencing independent of the speed of the OCP interface, while still taking advantage of synthesized synchronous design for maintaining a flexible packet format. Thus the NA leverages the advantages particular to either circuit design style. In Radulescu et al. [2004], a complete NA design for the ÆTHEREAL NoC is presented which also offers a shared-memory abstraction to the cores. It provides compatibility to existing on-chip protocols such as AXI, DTL, and OCP and allows easy extension to other future protocols as well.

However, the cost of using standard sockets is not trivial. As demonstrated in the HERMES NoC [Ost et al. 2005], the introduction of OCP makes the transactions up to 50% slower compared to the native core interface. An interesting design trade-off issue is the partitioning of the NA functions between software (possibly in the core) and hardware (most often in the NA). In Bhojwani and Mahapatra [2003], a comparison of software and hardware implementations of the packetization task was undertaken, the software taking 47 cycles to complete, while the hardware version took only 2 cycles. In Radulescu et al. [2004], a hardware implementation of the entire NA introduces a latency overhead of between 4 and 10 cycles, pipelined to maximize throughput. The NA in Bjerregaard et al. [2005] takes advantage of the low forward latency of clockless circuit techniques, introducing an end-to-end latency overhead of only 3 to 5 cycles for writes and 6 to 8 cycles for reads which include data return.

3.2. Network Level

The job of the network is to deliver messages from their source to their designated destination. This is done by providing the hardware support for basic communication primitives. A well-built network, as noted by Dally and Towles [2001], should appear as a logical wire to its clients. An on-chip network is defined mainly by its topology and the protocol implemented by it. Topology concerns the layout and connectivity of the nodes and links on the chip. Protocol dictates how these nodes and links are used.

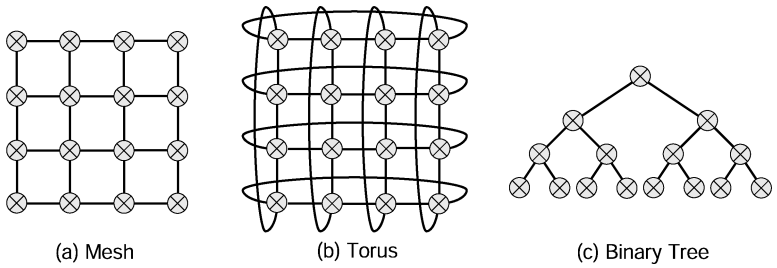

Fig. 9. Regular forms of topologies scale predictably with regard to area and power. Examples are (a) 4-ary 2-cube mesh, (b) 4-ary 2-cube torus and (c) binary (2-ary) tree.

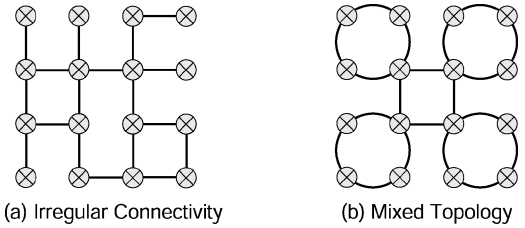

Fig. 10. Irregular forms of topologies are derived by altering the connectivity of a regular structure such as shown in (a) where certain links from a mesh have been removed or by mixing different topologies such as in (b) where a ring coexists with a mesh.

3.2.1. Topology. One simple way to distinguish different regular topologies is in terms of *k-ary n-cube* (grid-type), where k is the degree of each dimension and n is the number of dimensions (Figure 9), first described by Dally [1990] for multicomputer networks. The *k-ary tree* and the *k-ary n-dimensional* fat tree are two alternate regular forms of networks explored for NoC. The network area and power consumption scales predictably for increasing size of regular forms of topology. Most NoCs implement regular forms of network topology that can be laid out on a chip surface (a 2-dimensional plane) for example, k-ary 2-cube, commonly known as grid-based topologies. The Octagon NoC demonstrated in Karim et al. [2001, 2002] is an example of a novel regular NoC topology. Its basic configuration is a ring of 8 nodes connected by 12 bidirectional links which provides two-hop communication between any pair of nodes in the ring and a simple, shortest-path routing algorithm. Such rings are then connected edge-to-edge to form a larger, scalable network. For more complex structures such as trees, finding the optimal layout is a challenge on its own right.

Besides the form, the nature of links adds an additional aspect to the topology. In k-ary 2-cube networks, popular NoC topologies based on the nature of link are the mesh which uses bidirectional links and torus which uses unidirectional links. For a torus, a folding can be employed to reduce long wires. In the NOSTRUM NoC presented in Millberg et al. [2004], a folded torus is discarded in favor of a mesh with the argument that it has longer delays between routing nodes. Figure 9 shows examples of regular forms of topology. Generally, mesh topology makes better use of links (utilization), while tree-based topologies are useful for exploiting locality of traffic.

Irregular forms of topologies are derived by mixing different forms in a hierarchical, hybrid, or asymmetric fashion as seen in Figure 10. Irregular forms of topologies scale

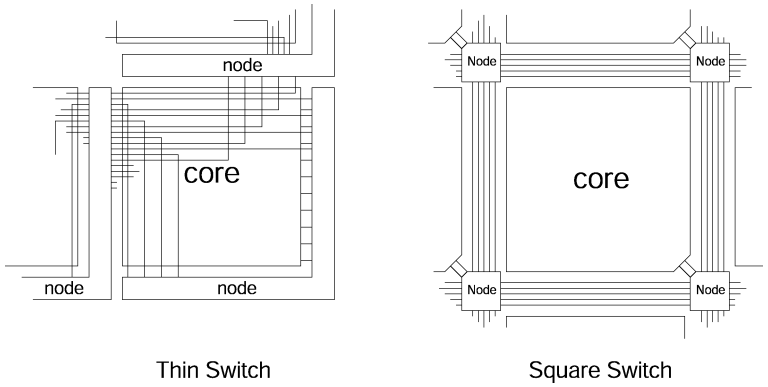

Fig. 11. Two layout concepts. The thin switch is distributed around the cores, and wires are routed across it. The square switch is placed on the crossings in dedicated channels between the cores.

nonlinearly with regards to area and power. These are usually based on the concept of clustering. A small private/local network often implemented as a bus [Mai et al. 2000; Wielage and Goossens 2002] for local communication with k-ary 2-cube global communication is a favored solution. In Pande et al. [2005], the impact of clustering on five NoC topologies is presented. It shows 20% to 40% reduction in bit-energy for the same amount of throughput due to traffic localization.

With regard to the presence of a local traffic source or sink connected to the node, *direct networks* are those that have at least one core attached to each node; *indirect networks*, on the other hand, have a subset of nodes not connected to any core, performing only network operations as is generally seen in tree-based topology where cores are connected at the leaf nodes. The examples of indirect tree-based networks are fat-tree in SPIN [Guerrier and Greiner 2000] and butterfly in Pande et al. [2003]. The fat-tree used in SPIN is proven in Leiserson [1985] to be most hardware efficient compared to any other network.

For alternate classifications of topology, the reader is referred to Aggarwal and Franklin [2002], Jantsch [2003], and Culler et al. [1998]. Culler et al. [1998] combine protocol and geometry to bring out a new type of classification which is defined as topology.

With regards to the routing nodes, a layout trade-off is the *thin switch* vs *square switch* presented by Kumar et al. [2002]. Figure 11 illustrates the difference between these two layout concepts. A thin switch is distributed around the cores, and wires are routed across them. A square switch is placed on the crossings of dedicated wiring channels between the cores. It was found that the square switch is better for performance and bandwidth, while the thin switch requires relatively low area. The area overhead required to implement a NoC is in any case expected to be modest. The processing logic of the router for a packet switched network is estimated to be approximately between 2.0% [Pande et al. 2003] to 6.6% [Dally and Towles 2001] of the total chip area. In addition to this, the wiring uses a portion of the upper two wiring layers.

3.2.2. Protocol. The protocol concerns the strategy of moving data through the NoC. We define switching as the mere transport of data, while routing is the intelligence behind it, that is, it determines the path of the data transport. This is in accordance with Culler et al. [1998]. In the following, these and other aspects of protocol commonly

addressed in NoC research, are discussed.

—*Circuit vs packet switching.* Circuit switching involves the circuit from source to destination that is setup and reserved until the transport of data is complete. Packet switched traffic, on the other hand, is forwarded on a per-hop basis, each packet containing routing information as well as data.

—*Connection-oriented vs connectionless.* Connection-oriented mechanisms involve a dedicated (logical) connection path established prior to data transport. The connection is then terminated upon completion of communication. In connectionless mechanisms, the communication occurs in a dynamic manner with no prior arrangement between the sender and the receiver. Thus circuit switched communication is always connection-oriented, whereas packet switched communication may be either connection-oriented or connectionless.

—*Deterministic vs adaptive routing.* In a deterministic routing strategy, the traversal path is determined by its source and destination alone. Popular deterministic routing schemes for NoC are source routing and X-Y routing (2D dimension order routing). In source routing, the source core specifies the route to the destination. In X-Y routing, the packet follows the rows first, then moves along the columns toward the destination or vice versa. In an adaptive routing strategy, the routing path is decided on a per-hop basis. Adaptive schemes involve dynamic arbitration mechanisms, for example, based on local link congestion. This results in more complex node implementations but offers benefits like dynamic load balancing.

—*Minimal vs nonminimal routing.* A routing algorithm is minimal if it always chooses among shortest paths toward the destination; otherwise it is nonminimal.

—*Delay vs loss.* In the delay model, datagrams are never dropped. This means that the worst that can happen is that the data is delayed. In the loss model, datagrams can be dropped. In this case, the data needs to be retransmitted. The loss model introduces some overhead in that the state of the transmission, successful or failed, must somehow be communicated back to the source. There are, however, some advantages involved in dropping datagrams, for example, as a means of resolving network congestion.

—*Central vs distributed control.* In centralized control mechanisms, routing decisions are made globally, for example, bus arbitration. In distributed control, most common for segmented interconnection networks, the routing decisions are made locally.

The protocol defines the use of the available resources, and thus the node implementation reflects design choices based on the listed terms. In Figure 12, taken from Duato et al. [2003], the authors have clearly identified the major components of any routing node that is, buffers, switch, routing and arbitration unit, and link controller. The switch connects the input buffers to the output buffers, while the routing and arbitration unit implements the algorithm that dictates these connections. In a centrally controlled system, the routing control would be common for all nodes, and a strategy might be chosen which guarantees no traffic contention. Thus no arbitration unit would be necessary. Such a scheme can be employed in a NoC in which all nodes have a common sense of time as presented in Millberg et al. [2004]. Here the NOSTRUM NoC implements an explicit time division multiplexing mechanism which the authors call *Temporally Disjoint Networks* (TDN). Packets cannot collide if they are in different TDNs. This is similar to the slot allocation mechanism in the ÆTHEREAL NoC [Goossens et al. 2005].

The optimal design of the switching fabric itself relates to the services offered by the router. In Kim et al. [2005], a crossbar switch is proposed which offers adaptive bandwidth control. This is facilitated by adding an additional bus, allowing the crossbar to be bypassed during periods of congestion. Thus, the switch is shown to improve the

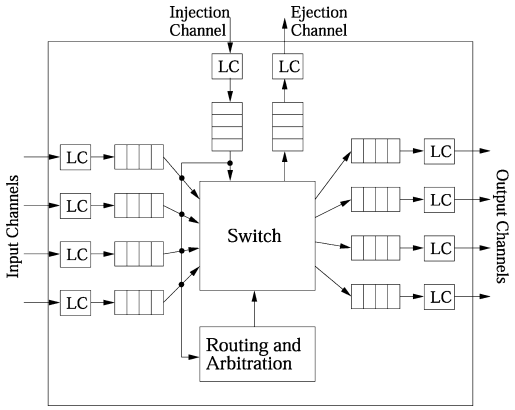

Fig. 12. Generic router model. LC = link controller (reprinted from Duato et al. [2003] by Jose Duato, Sudhakar Yalamanchili and Lionel Ni, Fig. 2.1, ©2003, with permission from Elsevier).

throughput and latency of the router by up to 27% and 41%, respectively, at a modest area and power overhead of 21% and 15%, respectively. In Bjerregaard and Sparsø [2005a], on the other hand, a nonblocking switch is proposed which allows for hard performance guarantees when switching connections within the router (more details in Section 3.2.4). By utilizing the knowledge that, only a limited number of flits can enter the router through each input port, the switch can be made to scale linearly rather than exponentially with the number of connections on each port. In Leroy et al. [2005], a switch similarly provides guaranteed services. This switch, however, switches individual wires on each port rather than virtual connections.

A quantitative comparison of connection-oriented and connectionless schemes for an MPEG-2 Video Decoder is presented in Harmanci et al. [2005]. The connection-oriented scheme is based on ÆTHEREAL, while the connectionless scheme is based on DiffServ—a priority-based packet scheduling NoC. The conclusions of tests, conducted in the presence of background traffic noise, show that (i) the individual end-to-end delay is lower in connectionless than in connection-oriented scheme due to better adaptation of the first approach to variable bit-rates of the MPEG video flows, and (ii) the connectionless schemes present a higher stability towards a wrong decision in the type of service to be assigned to a flow.

Concerning the merits of adaptive routing versus deterministic, there are different opinions. In Neeb et al. [2005], a comparison of deterministic (dimension-order) and adaptive (negative-first and planar-adaptive) routing applied to mesh, torus, and cube networks, was made. For chips performing interleaving in high throughput channel decoder wireless applications, the dimension-order routing scheme was found to be inferior compared to adaptive schemes when using lower dimension NoCs topologies. However, it was shown to be the best choice, due to low area and high thoughput characteristics, for higher dimension NoC topologies. The impact on area and throughput of input and output buffer queues in the router, was also discussed. In de Mello et al. [2004], the performance of minimal routing protocols in the HERMES [Moraes et al. 2004] NoC were investigated: one deterministic protocol (XY-routing) and three partially adaptive protocols (west-first, north-last and negative-first routing). While the adaptive protocols can potentially speed up the delivery of individual packets, it was shown that the deterministic protocol was superior to the adaptive ones from a global

Table II. Cost and Stalling for Different Routing Protocols

	Per router cost		
Protocol	Latency	Buffering	Stalling
store-and-forward	packet	packet	at two nodes and the link between them
wormhole	header	header	at all nodes and links spanned by the packet
virtual cut-through	header	packet	at the local node

point. The reason is that adaptive protocols tend to concentrate the traffic in the center of the network, resulting in increased congestion there.

The wide majority of NoC research is based on packet switching networks. In addition, most are delay-based since the overhead of keeping account of packets being transmitted and of retransmitting dropped packets is high. In Gaughan et al. [1996], however, a routing scheme is presented which accomodates droping packets when errors are detected. Most often connectionless routing is employed for best effort (BE) traffic (Section 4.3), while connection-oriented routing is used to provide service guarantees (Section 3.2.4). In SoCBUS [Sathe et al. 2003], a different approach is taken in that a connection-oriented strategy is used to provide BE traffic routing. Very simple routers establish short-lived connections set up using BE routed control packets which provide a very high throughput of 1.2GHz in a 0.18 μm CMOS process. Drawbacks are the time spent during the setup phase, which requires a path acknowledge, and the fact that only a single connection can be active on each link at any given time. A similarly connection-oriented NoC is aSoC [Liang et al. 2000] which implements a small *reconfigurable communication processor* in each node. This processor has interconnect memory that programs the crossbar for data transfer from different sources across the node on each communication cycle.

The most common forwarding strategies are store-and-forward, wormhole, and virtual cut-through. These will now be explained. Table II summarizes the latency penalty and storage cost in each node for each of these schemes.

Store-and-forward. Store-and-forward routing is a packet switched protocol in which the node stores the complete packet and forwards it based on the information within its header. Thus the packet may stall if the router in the forwarding path does not have sufficient buffer space. The CLICHE [Kumar et al. 2002] is an example of a store-and-forward NoC.

Wormhole. Wormhole routing combines packet switching with the data streaming quality of circuit switching to attain a minimal packet latency. The node looks at the header of the packet to determine its next hop and immediately forwards it. The subsequent flits are forwarded as they arrive. This causes the packet to *worm* its way through the network, possibly spanning a number of nodes, hence the name. The latency within the router is not that of the whole packet. A stalling packet, however, has the unpleasantly expensive side effect of occupying all the links that the worm spans. In Section 3.2.3, we will see how virtual channels can relieve this side effect at a marginal cost. In Al-Tawil et al. [1997], a well-structured survey of wormhole routing techniques is provided, and a comparison between a number of schemes is made.

Virtual cut-through. Virtual cut-through routing has a forwarding mechanism similar to that of wormhole routing. But before forwarding the first flit of the packet, the node waits for a guarantee that the next node in the path will accept the entire packet. Thus if the packet stalls, it aggregates in the current node without blocking any links.

While macronetworks usually employ store-and-forward routing, the prevailing scheme for NoC is wormhole routing. Advantages are low latency and the avoidance of

area costly buffering queues. A special case of employing single flit packets is explored in Dally and Towles [2001]. Here the data and header bits of the packets are transmitted separately and in parallel across a link, and the data path is quite wide (256 bits). Each flit is thus a packet in its own right, holding information about its destination. Hence, unlike wormhole routing, the stream of flits may be interlaced with other streams and stalling is restricted to the local node. Still single flit latency is achieved. The cost is a higher header-to-payload ratio, resulting in larger bandwidth overhead.

3.2.3. Flow Control. Peh and Dally [2001] have defined flow control as the mechanism that determines the packet movement along the network path. Thus it encompasses both global and local issues. Flow control mainly addresses the issue of ensuring correct operation of the network. In addition, it can be extended to also include issues on utilizing network resources optimally and providing predictable performance of communication services. Flow control primitives thus also form the basis of differentiated communication services. This will be discussed further in Section 3.2.4

In the following, we first discuss the concept of virtual channels and their use in flow control. We then discuss a number of works in the area, and, finally, we address buffering issues.

Virtual channels (VCs). VCs are the sharing of a physical channel by several logically separate channels with individual and independent buffer queues. Generally, between 2 and 16 VCs per physical channel have been proposed for NoC. Their implementation results in an area and possibly also power and latency overhead due to the cost of control and buffer implementation. There are however a number of advantageous uses. Among these are:

—*avoiding deadlocks.* Since VCs are not mutually dependent on each other, by adding VCs to links and choosing the routing scheme properly, one may break cycles in the resource dependency graph [Dally and Seitz 1987].

—*optimizing wire utilization.* In future technologies, wire costs are projected to dominate over transistor costs [ITRS 2003]. Letting several logical channels share the physical wires, the wire utilization can be greatly increased. Advantages include reduced leakage power and wire routing congestion.

—*improving performance.* VCs can generally be used to relax the interresource dependencies in the network, thus minimizing the frequency of stalls. In Dally [1992], it is shown that dividing a fixed buffer size across a number of VCs improve the network performance at high loads. In Duato and Pinkston [2001], the use of VCs to implement adaptive routing protocols is presented. Vaidya et al. [2001] and Cole et al. [2001] discusses the impact and benefit of supporting VCs.

—*providing diffentiated services.* Quality-of-service (QoS, see Section 3.2.4) can be used as a tool to optimize application performance. VCs can be used to implement such services by allowing high priority data streams to overtake those of lower priority [Felicijan and Furber 2004; Rostislav et al. 2005; Beigne et al. 2005] or by providing guaranteed service levels on dedicated connections [Bjerregaard and Sparsø 2005a].

To ensure correct operation, the flow control of the network must first and foremost avoid deadlock and livelock. Deadlock occurs when network resources (e.g., link bandwidth or buffer space) are suspended waiting for each other to be released, that is, where one path is blocked leading to other being blocked in a cyclic fashion [Dally and Seitz 1987]. It can be avoided by breaking cyclic dependencies in the resource dependency graph. Figure 13 illustrates how VCs can be used to prevent stalls due to dependencies on shared network resources. It is shown how in a network without VCs, stream B is

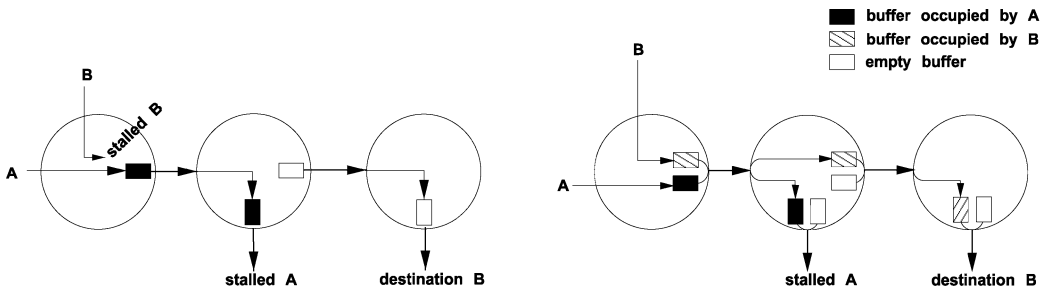

Fig. 13. Using virtual channels, independently buffered logical channels sharing a physical link, to prevent stalls in the network. Streams on different VCs can pass each other, while streams sharing buffer queues may stall.

stalled by stream A. In a network with VCs, however, stream B is assigned to a different VC with a separate buffer queue. Thus even though stream A is stalled stream B is enabled to pass.

Livelock occurs when resources constantly change state waiting for other to finish. Livelock is less common but may be expected in networks where packets are reinjected into the network or where backstepping is allowed, for example, during nonminimal adaptive routing.

Methods to avoid deadlock, and livelock can be applied either locally at the nodes with support from service primitives for example, implemented in hardware, or globally by ensuring logical separation of data streams by applying end-to-end control mechanisms. While local control is most widespread, the latter was presented in Millberg et al. [2004] using the concept of Temporally Disjoint Networks which was described in Section 3.2.2. As mentioned previously, dimension-ordered routing is a popular choice for NoC because it provides freedom from deadlock, without the need to introduce VCs. The *turn model* [Glass and Ni 1994] also does this but allows more flexibility in routing. A related approach is the *odd-even turn model* [Chiu 2000] for designing partially adaptive deadlock-free routing algorithms. Unlike the turn model, which relies on prohibiting certain turns in order to achieve freedom from deadlock, this model restricts the *locations* where some types of turns can be taken. As a result, the degree of routing adaptiveness provided is more even for different source-destination pairs. The ANoC [Beigne et al. 2005] implements this routing scheme.

The work of Jose Duato has addressed the mathematical foundations of routing algorithms. His main interests have been in the area of adaptive routing algorithms for multicomputer networks. Most of the concepts are directly applicable to NoC. In Duato [1993], the theoretical foundation for deadlock-free adaptive routing in wormhole networks is given. This builds on early work by Dally, which showed that by avoiding cyclic dependencies in the channel dependency graph of a network, deadlock-free operation is assured. Duato expands the theory to allow adaptive routing, and furthermore shows that the absence of cyclic dependencies is too restrictive. It is enough to require the existence of a channel subset which defines a connected routing subfunction with no cycles in its *extended* channel dependency graph. The extended channel dependency graph is defined in Duato [1993] as a graph for which the arcs are not only pairs of channels for which there is a direct dependency, but also pairs of channels for which there is an indirect dependency. In Duato [1995] and Duato [1996], this theory is refined and extended to cover also cut-through and store-and-forward routing. In Duato

and Pinkston [2001], a general theory is presented which glues together several of the previously proposed theories into a single theoretical framework.

In Dally and Aoki [1993], the authors investigated a hybrid of adaptive and deterministic routing algorithms using VCs. Packets are routed adaptively until a certain number of hops have been made in a direction away from the destination. Thereafter, the packets are routed deterministically in order to be able to guarantee deadlock-free operation. Thus the benefits of adaptive routing schemes are approached, while keeping the simplicity and predictability of deterministic schemes.

Other research has addressed flow control approaches purely for improving performance. In Peh and Dally [1999] and Kim et al. [2005], look-ahead arbitration schemes are used to allocate link and buffer access ahead of data arrival, thus reducing the end-to-end latency. This results in increased bandwidth utilization as well. Peh and Dally use virtual channels, and their approach is compared with simple virtual-channel flow control, as described in Dally [1992]. It shows an improvement in latency of about 15% across the entire spectrum of background traffic load, and network saturation occurs at a load 20% higher. Kim et al. do not use virtual channels. Their approach is shown to improve latency considerably (by 42%) when network load is low (10%) with much less improvement (13%) when network load is high (50%). In Mullins and Moore [2004], a virtual-channel router architecture for NoC is presented which optimizes routing latency by hiding control overheads, in a single cycle implementation.

Buffering. Buffers are an integral part of any network router. In by far the most NoC architectures, buffers account for the main part of the router area. As such, it is a major concern to minimize the amount of buffering necessary under given performance requirements. There are two main aspects of buffers (i) their size and (ii) their location within the router. In Kumar et al. [2002], it is shown that increasing the buffer size is not a solution towards avoiding congestion. At best, it delays the onset of congestion since the throughput is not increased. The performance improved marginally in relation to the power and area overhead. On the other hand, buffers are useful to absorb bursty traffic, thus leveling the bursts.

Tamir and Frazier [1988] have provided an comprehensive overview of advantages and disadvantages of different buffer configurations (size and location) and additionally proposed a buffering strategy called dynamically allocated multiqueue (DAMQ) buffer. In the argument of input vs. output buffers, for equal performance, the queue length in a system with output port buffering is always found to be shorter than the queue length in an equivalent system with input port buffering. This is so, since in a routing node with input buffers, a packet is blocked if it is queued behind a packet whose output port is busy (head-of-the-line-blocking). With regards to centralized buffer pools shared between multiple input and output ports vs distributed dedicated FIFOs, the centralized buffer implementations are found to be expensive in area due to overhead in control implementation and become bottlenecks during periods of congestion. The DAMQ buffering scheme allows independent access to the packets destined for each output port, while applying its free space to any incoming packet. DAMQ shows better performance than FIFO or statically-allocated shared buffer space per input-output port due to better utilization of the available buffer space especially for nonuniform traffic. In Rijpkema et al. [2001], a somewhat similar concept called virtual output queuing is explored. It combines moderate cost with high performance at the output queues. Here independent queues are designated to the output channels, thus enhancing the link utilization by bypassing blocked packets.

In Hu and Marculescu [2004a], the authors present an algorithm which sizes the (input) buffers in a mesh-type NoC on the basis of the traffic characteristics of a given

application. For three audio/video benchmarks, it was shown how such intelligent buffer allocation resulted in about 85% savings in buffering resources in comparison to uniform buffer sizes without any reduction in performance.

3.2.4. Quality of Service (QoS). QoS is defined as service quantification that is provided by the network to the demanding core. Thus it involves two aspects: (i) defining the services represented by a certain quantification and (ii) negotiating the services. The services could be low latency, high throughput, low power, bounds on jitter, etc. Negotiating implies balancing the service demands of the core with the services available from the network.

In Jantsch and Tenhunen [2003, 61–82], Goossens et al characterize the nature of QoS in relation to NoC. They identify two basic QoS classes, best-effort services (BE) which offer no commitment, and guaranteed services (GS) which do. They also present different levels of commitment, and discuss their effect on predictability of the communication behavior: 1) correctness of the result, 2) completion of the transaction, 3) bounds on the performance. In Rijpkema et al. [2001], argumentation for the necessity of a combination of BE and GS in NoC is provided. Basically, GS incur predictability, a quality which is often desirable, for example, in real-time systems, while BE improves the average resource utilization [Jantsch and Tenhunen 2003, 61–82; Goossens et al. 2002; Rijpkema et al. 2003]. More details of the advantages of GS from a design flow and system verification perspective are given in Goossens et al. [2005] in which a framework for the development of NoC-based SoC, using the ÆTHEREAL NoC, is described.

Strictly speaking, BE refers to communication for which no commitment can be given whatsoever. In most NoC-related works, however, BE covers the traffic for which only correctness and completion are guaranteed, while GS is traffic for which additional guarantees are given, that is, on the performance of a transaction. In macronetworks, service guarantees are often of a statistical nature. In tightly bound systems such as SoC, hard guarantees are often preferred. GS allows analytical system verification, and hence a true decoupling of subsystems. In order to give hard guarantees, GS communication must be logically independent of other traffic in the system. This requires connection-oriented routing. Connections are instantiated as virtual circuits which use logically independent resources, thus avoiding contention. The virtual circuits can be implemented by either virtual channels, time-slots, parallel switch fabric, etc. As the complexity of the system increases and as GS requirements grow, so does the number of virtual circuits and resources (buffers, arbitration logic, etc) needed to sustain them.

While hard service guarantees provide an ultimate level of predictability, soft (statistical) GS or GS/BE hybrids have also been the focus of some research. In Bolotin et al. [2004], Felicijan and Furber [2004], Beigne et al. [2005] and Rostislav et al. [2005], NoCs providing prioritized BE traffic classes are presented. SoCBUS [Sathe et al. 2003] provides hard, short-lived GS connections; however, since these are setup using BE routed packets, and torn down once used, this can also be categorized as soft GS.

ÆTHEREAL [Goossens et al. 2005], NOSTRUM [Millberg et al. 2004], MANGO [Bjerregaard and Sparsø 2005a], SONICS [Weber et al. 2005], aSOC [Liang et al. 2004], and also the NoCs presented in Liu et al. [2004], in Leroy et al. [2005], and the static NoC used in the RAW multiprocessor architecture [Taylor et al. 2002], are examples of NoCs implementing hard GS. While most NoCs that implement hard GS use variants of time division multiplexing (TDM) to implement connection-oriented packet routing, thus guaranteeing bandwidth on connections, the clockless NoC MANGO uses sequences of virtual channels to establish virtual end-to-end connections. Hence limitations of TDM, such as bandwidth and latency guarantees which are inversely proportional, can be overcome by appropriate scheduling. In Bjerregaard and Sparsø [2005b], a scheme

for guaranteeing latency, independently of bandwidth, is presented. In Leroy et al. [2005], an approach for allocating individual wires on the link for different connections is proposed. The authors call this *spatial division multiplexing* as opposed to TDM.

For readers interested in exploitation of GS (in terms of throughput) virtual circuits during idle times, in Andreasson and Kumar [2004, 2005] the concept of slack-time aware routing is introduced. A producer manages injection of BE packets during the slacks in time-slots reserved for GS packets, thereby mixing GS and BE traffic at the source which is unlike other schemes discussed so far where it is done in the routers. In Andreasson and Kumar [2005], the impact of variation of output buffer on BE latency is investigated, while in Andreasson and Kumar [2004], the change of injection control mechanism for fixed buffer size is documented. QoS can also be handled by controlling the injection of packets into a BE network. In Tortosa and Nurmi [2004], scheduling schemes for packet injection in a NoC with a ring topology were investigated. While a basic scheduling, which always favors traffic already in the ring, provided the highest total bandwidth, weighted scheduling schemes were much more fair in their serving of different cores in the system.

In addition to the above, QoS may also cover special services such as:

—*broadcast, multicast, narrowcast.* These features allow simultaneous communication from one source to all, that is, broadcast, or select destinations as is shown in ÆTHEREAL [Jantsch and Tenhunen 2003, 61–82] where a master can perform read or write operations on an address-space distributed among many slaves. In a connection-oriented environment, the master request is channeled to a single slave for execution in narrowcast, while the master request is replicated for execution at all slaves in multicast. APIs are available within the NA to realize these types of transactions [Radulescu et al. 2004]. An alternate mulitcast implementation is discussed in Millberg et al. [2004] where a virtual circuit meanders through all the destinations.

—*virtual wires.* This refers to the use of network message-passing services to emulate direct pin-to-pin connection. In Bjerregaard et al. [2005], such techniques are used to support a flexible interrupt scheme in which the interrupt of a slave core can be programmed to trigger any master attached to the network by sending a trigger packet.

—*complex operations.* Complex functionality such as test-and-set issued by a single command across the network can be used to provide support for, for example, semaphores.

3.3. Link Level

Link-level research regards the node-to-node links. These links consist of one or more channels which can be either virtual or physical. In this section, we present a number of areas of interest for link level research: synchronization, implementation, reliability, and encoding.

3.3.1. Synchronization. For link-level synchronization in a multiclock domain SoC, Chelcea and Nowick [2001] have presented a mixed-time FIFO design. The FIFO employs a ring of storage elements in which tokens are used to indicate full or empty state. This simplifies detection of the state of the FIFO (full or empty) and thus makes synchronization robust. In addition, the definitions of full and empty are extended so that full means that 0 or 1 cell is unused, while empty means only 0 or 1 cells is used. This helps in hiding the synchronization delay introduced between the state detection and the input/output handshaking. The FIFO design introduced can be made arbitrarily robust with regards to metastability as settling time and latency can be traded off.

With the emerging of the GALS concept of *globally asynchronous locally synchronous* systems [Chapiro 1984; Meincke et al. 1999], implementing links using asynchronous circuit techniques [Sparsø and Furber 2001; Hauck 1995] is an obvious possibility. A major advantage of asynchronous design styles relevant for NoC is the fact that, apart from leakage, no power is consumed when the links are idle. Thus, the design style also addresses the problematic issue of increasing power usage by large chips. Another advantage is the potentially very low forward latency in uncongested data paths leading to direct performance benefits. Examples of NoCs based on asynchronous circuit techniques are CHAIN [Bainbridge and Furber 2002; Amde et al. 2005], MANGO [Bjerregaard and Sparsø 2005a], ANoC [Beigne et al. 2005], and QNoC [Rostislav et al. 2005]. Asynchronous logic incorporates some area and dynamic power overhead compared with synchronous logic due to local handshake control. The 1-of-4 encodings discussed in Section 3.3.4, generalized to 1-of-N, is often used in asynchronous links [Bainbridge and Furber 2001].

On the other hand, resynchronization of an incoming asynchronous transmission is also not trivial. It costs both time and power, and bit errors may be introduced. In Dobkin et al. [2004], resynchronization techniques are described, and a method for achieving high throughput across an asynchronous to synchronous boundary is proposed. The work is based on the use of stoppable clocks, a scheme in which the clock of a core is stopped while receiving data on an asynchronous input port. Limitations to this technique are discussed, and the proposed method involves only the clock on the input register being controlled. In Ginosaur [2003], a number of synchronization techniques are reviewed, and the pitfalls of the topic are addressed.

The trade-offs in the choice of synchronization scheme in a globally asynchronous or multiclocked system is sensitive to the latency requirements of the system, the expected network load during normal usage, the node complexity, etc.

3.3.2. Implementation Issues. As chip technologies scale into the DSM domain, the effect of wires on link delays and power consumption increase. Aspects and effects on wires of technology scaling are presented in Ho et al. [2001], Lee [1998], Havemann and Hutchby [2001], and Sylvester and Keutzer [2000]. In Liu et al. [2004], these issues are covered specifically from a NoC point-of-view, projecting the operating frequency and size of IP cores in NoC-based SoC designs for future CMOS technologies down to 0.05 μm. In the following, we will discuss a number of physical level issues relevant to the implementation of on-chip links.

Wire segmentation. At the physical level, the challenge lies in designing fast, reliable and low power point-to-point interconnects, ranging across long distances. Since the delay of long on-chip wires is characterized by distributed RC charging, it has been standard procedure for some time to apply segmentation of long wires by inserting *repeater* buffers at regular intervals in order to keep the delay linearly dependent on the length of the wire. In Dobbelaere et al. [1995], an alternative type of repeater is proposed. Rather than splitting and inserting a buffer in the path of the wire, it is based on a scheme of sensing and pulling the wire using a keeper device attached to the wire. The method is shown to improve the delay of global wires by up to 2 times compared with conventional repeaters.

Pipelining. Partitioning long interconnects into pipeline stages as an alternative to wire segmentation is an effective way of increasing throughput. The flow control handshake loop is shorter in a pipelined link, making the critical loop faster. This is at the expense of latency of the link and circuit area since pipeline stages are more complex than repeater buffers. But the forward latency in an asynchronous pipeline handshake cycle can be minimized to a few gate delays so, as wire effects begin to dominate performance

in DSM technologies, the overhead of pipelining as opposed to buffering will dwindle. In Singh and Nowick [2000], several high-throughput clockless pipeline designs were implemented using dynamic logic. Completion detection was employed at each stage to generate acknowledge signals which were then used to control the precharging and evaluation of the dynamic nodes. The result was a very high throughput of up to 1.2 GDI/s (giga data items per second) for single rail designs, in a 0.6 μm CMOS technology. In Mizuno et al. [2001], a hybrid of wire segmentation and pipelining was shown in that a channel was made with segmentation buffers implemented as latches. A congestion signal traveling backwards through the channel compresses the data in the channel, storing it in the latches until the congestion is resolved. Thus a back pressure flow control scheme was employed without the cost of full pipeline elements.

Low swing drivers. In an RC charging system, the power consumption is proportional to the voltage shift squared. One way of lowering the power consumption for long on-chip interconnects is by applying low-swing signaling techniques which are also widely used for off-chip communication lines. Such techniques are presented and analyzed in Zhang et al. [1999]. Basically the power usage is lowered at the cost of the noise margin. However, a differential transmission line (2 wires), on which the voltage swing is half that of a given single-ended transmission line, has differential mode noise characteristics comparable to the single-ended version. This is so because the voltage difference between the two wires is the same as that between the single-ended wire and a mid-point between supply and ground. As an approximation, it uses only half the power, however, since the two wires working at half the swing each consume one-fourth the power. The common mode noise immunity of the differential version is also greatly improved, and it is thus less sensitive to crosstalk and ground bounces, important sources of noise in on-chip environments as discussed in the reliability section that follow. In Ho et al. [2003], the design of a low-swing, differential on-chip interconnect for the Smart Memories [Mai et al. 2000] is presented and validated with a test chip.

In Svensson [2001] the author demonstrated how an optimum voltage swing for minimum power consumption in on- and off-chip interconnects can be found for a given data activity rate. The work takes into account dynamic and static power consumption of driving the wire as well as in the receiver, which needs to amplify the signal back to full logic level. Calculations are presented for a 0.18 μm CMOS technology. Figure 14 displays the power consumption versus voltage swing for a global on-chip wire of 5–10 mm, a power supply of 1.3 V, and a clock frequency of 1 GHz. For a data activity rate of 0.25 (random data), it is shown that there is a minimum at 0.12 V. This minimum occurs for a two-stage receiver amplifier and corresponds to a power saving of 17x. Using a single stage amplifier in the receiver, there is a minimum at 0.26 V, corresponding to a power saving of 14x.

Future issues. In Heiliger et al. [1997], the use of microstrip transmission lines as waveguides for sub-mm wave on-chip interconnects is analyzed. It is shown that using SiO_2 as dielectric exhibits prohibitively large attenuation. However, the use of bisbenzocyclobutene-polymer offers favorable line parameters, with an almost dispersion free behavior at moderate attenuation ($\leq$ 1 dB/mm at 100 GHz). In Kapur and Saraswat [2003], a comparison between electrical and optical interconnects for on-chip signaling and clock distribution is presented. Figure 15 shows the models used in evaluating optical and electrical communication. The delay vs. power and delay vs. interconnect length trade-offs are analyzed for the two types of signaling. In Figure 16, it is shown that the critical length above which the optical system is faster than the electrical one is approximately 3–5 mm, projected for a 50 nm CMOS fabrication technology with copper wiring. The work also shows that, for long interconnects (defined as 10 mm and above), the optical communication has a great potential for low power

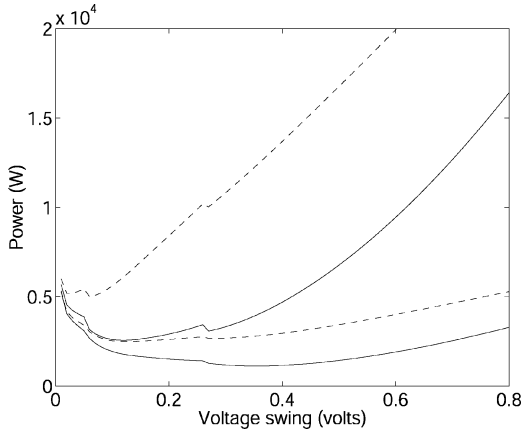

Fig. 14. Total power versus voltage swing for long (5–10 mm) on-chip interconnect. Solid line case 1: power supply generated off-chip by high efficiency DC-DC converter. Dashed line case 2: power supply generated internally on-chip. Upper curves for data activity of 0.25, lower curves 0.05 (reprinted from Svensson [2001] Fig. 2, ©2001 with permission from Christer Svensson).

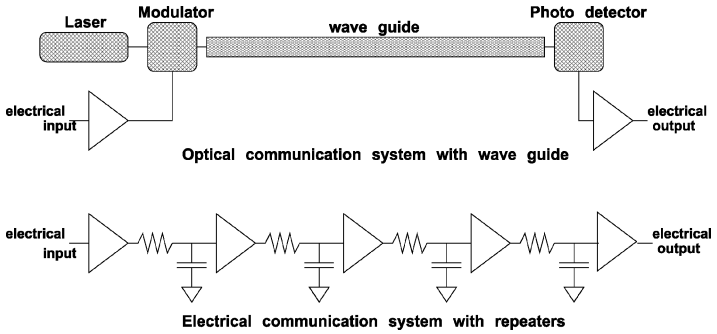

Fig. 15. Model of electrical and optical signaling systems for on-chip communication, showing the basic differences.

operation. Thus it is projected to be of great use in future clock distribution and global signaling.

3.3.3. Reliability. Designing global interconnects in DSM technologies, a number of communication reliability issues become relevant. Noise sources which can have an influence on this are mainly crosstalk, power supply noise such as ground bounce, electromagnetic interference (EMI), and intersymbol interference.

Crosstalk is becoming a serious issue due to decreasing supply voltage, increasing wire to wire capacitance, increasing wire inductance (e.g., in power supply lines), and increasing rise times of signaling wires. The wire length at which the peak crosstalk voltage is 10% of the supply voltage decreases drastically with technology scaling [Jantsch and Tenhunen 2003, chap. 6], and, since the length of global interconnects does not scale with technology scaling, this issue is especially relevant to the implementation of

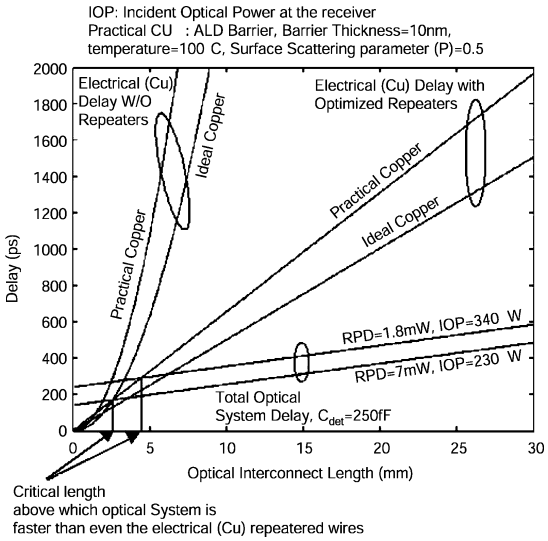

Fig. 16. Delay comparison of optical and electrical interconnect (with and without repeaters) in a projected 50 nm technology (reprinted from Kapur and Saraswat [2003] by Pawan Kapur and Krishna C. Saraswat, Fig. 13, ©2002, with permission from Elsevier).

NoC links. Power supply noise is worsened by the inductance in the package bonding wires, and the insufficient capacitance in the on-chip power grid. The effect of EMI is worsening as the electric charges moved by each operation in the circuit is getting smaller, making it more susceptible to external influence. Intersymbol interference, that is, the interference of one symbol on the following symbol on the same wire, is increasing as circuit speeds go up.

In Jantsch and Tenhunen [2003, chap. 6], Bertozzi and Benini present and analyze a number of error detecting/correcting encoding schemes in relation to NoC link implementation. Error recovery is a very important issue, since an error in, for instance, the header of a packet, may lead to deadlock in the NoC, blocking the operation of the entire chip. This is also recognized in Zimmer and Jantsch [2003] in which a fault model notation is proposed which can represent multiwire and multicycle faults. This is interesting due to the fact that crosstalk in DSM busses can cause errors across a range of adjacent bits. It is shown that, by splitting a wide bus into separate error detection bundles, and interleaving these, the error rate after using single-error correcting and double-error detecting codes can be reduced by several orders of a magnitude. This is because these error-correction schemes function properly when only one or two errors occur in each bundle. When the bundles are interleaved, the probability of multiple errors within the same bundle is greatly reduced.

In Gaughan et al. [1996] the authors deal with dynamically occurring errors in networks with faulty links. Their focus is on routing algorithms that can accommodate such errors, assuming that support for the detection of the errors is implemented. For wormhole routing, they present a scheme in which a data transmission is terminated upon detection of an error. A *kill* flit is transmitted backwards, deleting the worm and telling the sender to retransmit it. This naturally presents an overhead and is not generally representative for excising NoC implementations. It can, however, prove necessary in mission critical systems. The paper provides formal mathematical proofs of deadlock-freedom.

Another issue with new CMOS technologies is the fact that the delay distribution—due to process variations—flattens with each new generation. While typical delay improves, worst case delay barely changes. This presents a major problem in todays design methodologies as these are mostly based on worst case assumptions. Self-calibrating methods, as used in Worm et al. [2005], are a way of dealing with unreliability issues of this character. The paper presents a self-calibrating link, and the problem of adaptively controlling its voltage and frequency. The object is to maintain acceptable design trade-offs between power consumption, performance, and reliability when designing on-chip communication systems using DSM technologies.

Redundant transmission of messages in the network is also a way of dealing with fabrication faults. In Pirretti et al. [2004], two different flooding algorithms and a random walk algorithm are compared. It is shown that the flooding algorithms have an exceedingly large communication overhead, while the random walk offers reduced overhead and still maintains useful levels of fault tolerance.

With the aim of improving fabrication yield, Dally and Towles [2001] propose extra wires between nodes so that defective wires found during postproduction tests or during self-test at start-up can be bypassed. Another potential advantage of a distributed shared communication structure is the possibility of bypassing entire regions of a chip if fabrication faults are found.

Dynamic errors are more likely in long wires and segmenting links into pipeline stages helps to keep the error rate down and the transmission speed up. Since segmentation of the communication infrastructure is one of the core concepts of NoC, it inherently provides solutions to the reliability problems. The segmentation is made possible because NoC-based systems generally imply the use of programming models allowing some degree of latency insensitive communication. Thus it is shown how the issues and solutions at the physical level relate directly to issues and solutions at system level, and vice versa. Another solution towards avoiding dynamic errors is the shielding of signal wires, for example, by ground wires. This helps to minimize crosstalk from locally interfering wires at the expense of wiring area.

3.3.4. Encoding. Using encoding for on-chip communication has been proposed; the most common objective is to reduce power usage per communicated bit, while maintaining high speed and good noise margin. In Bogliolo [2001], the proposed encoding techniques are categorized as speed-enhancing or low-power encodings, and it is shown how different schemes in these two categories can be combined to gain the benefits of both. In Nakamura and Horowitz [1996], a very simple low-weight coding technique was used to reduce dI/dt noise due to simultaneous switching of off-chip I/O drivers. An 8-bit signal was simply converted to a 9-bit signal, the 9th bit indicating whether the other 8 bits should be inverted. The density of 1's was thus reduced, resulting in a reduction of switching noise by 50% and of power consumption by 18%. Similar techniques could prove useful in relation to long on-chip interconnects. The abundant wire resources available on-chip can also be used to implement more complex M-of-N encodings, thus trading wires for power. A widely used technique, especially in asynchronous implementations, is 1-of-4 encoding. This results in a good power/area trade-off and low encoding/decoding overhead [Bainbridge and Furber 2001; Bainbridge and Furber 2002].

Another area of encoding, also discussed in Section 3.3.3, relates to error management. This involves the detection and correction of errors that may occur in the network. The mechanism may be observed at different layers of the network and thus be applicable to either phits, flits, packets, or messages. With regards to NoC, the interesting issues involve errors in the links connecting the nodes since long wires

of deep submicron technologies may exhibit unreliable behavior (see Section 3.3.3). ×pipes [Osso et al. 2003] implements a flit-level CRC mechanism, running in parallel with switching (thus masking its delay), to detect errors. Another common technique is parity-checks. The need here is to balance complexity of error-correction circuits to the urgency of such mechanisms.

An interesting result is obtained in Jantsch and Vitkowski [2005] wherein the authors investigate power consumption in the NOSTRUM NoC. Results are based on a 0.18 μm implementation and scaled down to 65 nm. The paper concludes that the major part of the power is spent in the link wires. Power-saving encoding however reduces performance and simply scaling the supply voltage to normalize performance—in nonencoded links—actually results in better power figures than any of the encoding schemes investigated. Subsequently, the authors propose the use of end-to-end data protection through error correction methods which allows voltage scaling, while maintaining the fault probability without lowering the link speed. In effect, this results in better power figures.

In this section, we have discussed issues relevant to the lowest level of the NoC, the link level. This concludes the discussion of network design and implementation topics. In the following section, we discuss NoC from the view of design approach and modeling in relation to SoC.

4. NOC MODELING

NoC, described as a subset of SoC, is an integral part of SoC design methodology and architecture. Given the vast design space and implementation decisions involved in NoC design, modeling and simulation is important to design flow, integration, and verification of NoC concepts. In this section, we first discuss issues related to NoC modeling, and then we explore design methodology used to study the system-level impact of the NoC. Finally, traffic characterization, which bridges system-level dynamics with NoC requirements, is discussed.

4.1. Modeling

Modeling the NoC in abstract software models is the first means to approach and understand the required NoC architecture and the traffic within it. Conceptually the purpose of NoC modeling is (i) to explore the vast design and feature space, and (ii) to evaluate trade-offs between power, area, design-time, etc; while adhering to application requirements on one side and technology constraints on the other side. Modeling NoC has three intertwined aspects: modeling environment, abstraction levels, and result analysis. In the modeling environment section, we present three frameworks to describe NoC. Section 4.1.2 discusses work done across different levels of NoC abstraction. The result analysis deals with a wide range of issues and is hence dealt with separately in Section 5.

4.1.1. Modeling Environment. The NoC models are either analytical or simulation based and can model communication across abstractions.

In a purely abstract framework, a NoC model using allocators, scheduler, and synchronizer is presented in Madsen et al. [2003] and Mahadevan et al. [2005]. The allocator translates the path traversal requirements of the message in terms of its resource requirements such as bandwidth, buffers, etc. It attempts to minimize resource conflicts. The scheduler executes the message transfer according to the particular network service requirements. It attempts to minimize resource occupancy. A synchronizer models the dependencies among communicating messages allowing concurrency. Thus these

three components are well suited to describe a wide variety of NoC architectures and can be simulated in a multiprocessor real-time environment.

OPNET, a commercial network simulator originally developed for macronetworks, is used as a NoC simulator in Bolotin et al. [2004], Xu et al. [2004], and Xu et al. [2005]. OPNET provides a convenient tool for hierarchical modeling of a network, including processes (state machines), network topology description, and simulation of different traffic scenarios. However, as noted in Xu et al. [2004] and Xu et al. [2005], it needs to be adapted for synchronous environments, requiring explicit design of clocking scheme and a distribution network. Bolotin et al. [2004] uses OPNET to model a QoS-based NoC architecture and design with irregular network topology.

A VHDL-based cycle accurate RTL model for evaluating power and performance of NoC architecture is presented in Banerjee et al. [2004]. The power and delay are evaluated for fine-grain components of the routers and links using SPICE simulations for a 0.18 μm technology and incorporated into the architectural-level blocks. Such modeling enables easy evaluation of dynamic vs leakage power at the system level. As expected, at high injection rate (packets/cycle/node), it was found that dynamic power dominates over leakage power. The Orion power performance simulator proposed by Wang et al. [2002] modeled only dynamic power consumption.

Recently, due to the increasing size of applications, NoC emulation [Genko et al. 2005] has been proposed as an alternative to simulation-based NoC models. It has been shown that FPGA-based emulation can take a few seconds compared to simulation-based approaches which can take hours to process through many millions of cycles as would be necessary in any thorough communication coexploration.

4.1.2. Noc Modeling at Different Abstraction Levels. New hardware description languages are emerging, such as SystemC [2002], a library of C++, and SystemVerilog [Fitzpatrick 2004], which make simulations at a broad range of abstraction levels readily available and thus support the full range of abstractions needed in a modular NoC-based SoC design. In Bjerregaard et al. [2004], mixed-mode asynchronous handshake channels were developed in SystemC, and a mixed abstraction-level design flow was used to design two different NoC topologies.

From an architectural point of view, the network topology generally incur the use of a segmented (multihop) communication structure, however, some researchers, working at the highest levels of abstraction, define NoC merely as a multiport blackbox communication structure or core, presenting a number of ports for communication. A message can be transmitted from an arbitrary port to any other, allowing maximum flexibility of system communication. At this level, the actual implementation of the NoC is often not considered. Working at this high abstraction level allows a great degree of freedom from lower level issues. Table III adapted from Gerstlauer [2003] summarizes, in general, the communication primitives at different levels of abstraction.

At system level, transaction-level models (TLM) are typically used for modeling communication behavior. This takes the form of either synchronous or asynchronous *send()*/ *receive()* message passing semantics which use unique channels for communication between the source and the destination. One level below this abstraction, for NoCs, additional identifiers such as addressing may be needed to uniquely identify the traversal path or for providing services for end-to-end communication. Control primitives at network and link level, which are representative of actual hardware implementation, model the NoC flow-control mechanisms. In Gerstlauer [2003], a JPEG encoder and voice encoder/decoder running concurrently were modeled for each and for mixed levels of abstraction. The results show that the model complexity generally grows exponentially with a lower level of abstraction. By extrapolating the result from bus to NoC, interestingly, model complexity at NA level can be found to be higher than at other

Table III. Communication Semantics and Abstraction for NoC, Aadapted From Gerstlauer [2003]

Layer	Interface semantics	Communication

Application/ Presentation	IP-to-IP messaging
	`sys.send(struct myData)`
	`sys.receive(struct myData)`
		Message passing

Session/ Transport	IP-to-IP port-oriented messaging
	`nwk.read(messagepointer*, unsigned len)`
	`nwk.write(int addr, msgptr*, unsigned len)`
		Message passing or shared memory

Network	NA-to-NA packet streams
	`ctrl.send(), ctrl.receive()`
	`link.read(bit[] path, bit[] data_packet)`
	`link.write(bit[] path, bit[] data_packet)`
		Message passing or shared memory

Link	Node-to-Node logical links and shared byte streams
	`ctrl.send(), ctrl.receive()`
	`channel.transmit(bit[] link, bit[] data_flit)`
	`channel.receive(bit[] link, bit[] data_flit)`
		Message passing

Physical	Pins and wires `A.drive(0), D.sample(), clk.tick()`	Interconnect

levels due to the slicing of message, connection management, buffer management, and others.

Working between a session to network layer, Juurlink and Wijshoff [1998] have made a comparison of three communication models used in parallel computation: (i) asynchronous communication with fixed message size, (ii) synchronous communication which rewards burst-mode message transfers, and (iii) asynchronous with variable message size communication while also accounting for network load. Cost-benefit analysis shows that, though the software-based messaging layers serve a very useful function of delinking computation and communication, it creates anywhere from between 25% to 200% overhead as opposed to optimized hardware implementation.

A similar study of parallel computation applications, but with a more detailed network model, was undertaken by Vaidya et al. [2001]. Here the network was implemented to use adaptive routing with virtual channels. The applications, running on power-of-two number of processors using grid-based network topologies, used shared memory or message passing for communication, thus generating a wide range of traffic patterns. They found that increasing the number of VCs and routing adaptively offers little performance improvement for scalable shared memory applications. Their observation holds true over a range of systems and problem sizes. The results show that the single most important factor for improving performance in such applications is the router speed which is likely to provide lasting payoffs. The benefits of a faster router are visible across all applications in a consistent and predictable fashion.

Ahonen et al. [2004] and Lahiri et al. [2001] have associated high-level modeling aspects with actual design choices, such as selection of an appropriate topology, selection of communication protocols, specification of architectural parameters (such as bus widths, burst transfer size, priorities, etc), and mapping communications onto the architecture, as requirements to optimize the on-chip communication for application-specific needs. Using a tool called OIDIPUS, Ahonen et al. [2004] compare placement of twelve processors in a ring-based topology. They found that OIDIPUS, which uses the physical path taken by the communication channels as the cost function, generated topologies that are only marginally inferior to human design. Without being restricted to any one topology, Lahiri et al. [2001] have evaluated traffic characteristics in a static priority-based shared bus, hierarchical bus, two-level time division multiplexed access (TDMA), and ring-based communication architecture. They found that no single architecture uniformly outperforms other.

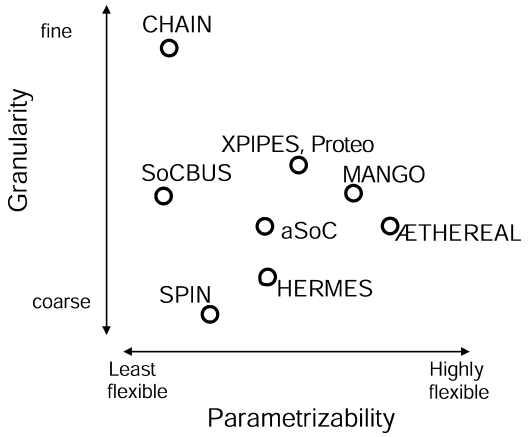

Fig. 17. NoC instantiation space.

Wieferink [2004] have demonstrated a processor/communication coexploration methodology which works cross-abstraction and in a cosimulation platform. Here LISA-based IP core descriptions have been integrated with SystemC-based bus-based transaction-level models. A wide range of APIs are then provided to allow modeling between LISA and SystemC models, to allow instruction accurate models to coexist with cycle accurate models and TLM with RTL models. MPARM [Loghi et al. 2004] is a similar cycle accurate and SystemC coexploration platform used in exploration of AMBA, STBus, and xpipes NoC evaluation.

4.2. Design and Coexploration Methodology

The NoC components, as described in Section 2.1, lends itself to flexible NoC designs such as parameterizable singular IP core or malleable building blocks, customizable at the network layer for design and reuse into application-specific NoC. A SoC design methodology requiring a communication infrastructure can exploit either characteristics to suit the application's needs. Keeping this in mind, different NoC researchers have uniquely tailored their NoC architectures. Figure 17 shows our assessment of instance-specific capability of these NoC architectures. The two axis are explained as follows.

—*Parametrizability at system-level.* By this, we mean the ease with which a system-level NoC characteristic can be changed at instantiation time. The NoC description may encompass a wide range of parameters, such as: number of slots in the switch, pipeline stages in the links, number of ports of the network, and others. This is very useful for coexploration directly with IP cores of the SoC.

—*Granularity of NoC.* By granularity, we mean at what level the NoC or NoC components is described. At the coarser end, the NoC may be described as a single core, while at the other end of the spectrum, the NoC may be assembled from lower-level blocks.

Consider the example of CHAIN [Bainbridge and Furber 2002]. It provides a library of fine-grained NoC components. Using these components, a NoC designer can use a Lego-brick approach to build the desired NoC topology, though as system-level block such a NoC has low flexibility. Thus it may be disadvantageous when trying to find the optimum SoC communication architecture in a recursive design space exploration

process. The ÆTHEREAL [Goossens et al. 2002], SoCBUS [Sathe et al. 2003], and aSoC [Liang et al. 2000] networks describe the NoC as a relatively coarse grain system-level module but with widely different characteristics. The ÆTHEREAL is highly flexible in terms of adjusting the available slots, number of ports, etc., which is useful for NoC exploration; whereas aSoC and SoCBUS do not expose many parameters for change (though aSoC supports flexible programming of connections after instantiation). The SPIN NoC [Guerrier and Greiner 2000], designed as a single IP core, is least parameterizable with its fixed topology and protocol. Interestingly, the xpipes [Osso et al. 2003] provides not merely a set of relatively fine-grain soft-macros of switches and pipelined links which the xpipesCompiler [Jalabert et al. 2004] uses to automatically instantiate an application-specific network, but also enables optimization of system-level parameters such as removing redundant buffers from output ports of switches, sharing signals common to objects, etc. This lends itself to both flexibility for coexploration and easy architectural changes when needed. Similarly, conclusions can be drawn of Proteo [Siguenza-Tortosa et al. 2004], HERMES [Moraes et al. 2004] and MANGO [Bjerregaard and Sparsø 2005a] NoCs. A detailed comparison of different features of most of the listed NoCs is tabulated in Moraes et al. [2004].

The impact on SoC design time and coexploration of different NoC design styles listed is considerable. For example, in Jalabert et al. [2004], during design space exploration, to find an optimum NoC for three video applications, that is, video object plane decoder, MPEG4 decoder and multiwindow displayer, the xpipesCompiler found that irregular networks with large switches may be more advantageous than regular networks. This is easier to realize in a macroblock NoC such as CHAIN or xpipes than it is in NoC designed as a single (system level) IP core such as SPIN. The basis for the compiler's decision is the pattern of traffic generated by the application. This is the focus of the next section. Further explanation of trade-offs in using a flexible instantiation-specific NoC can be found in Pestana et al. [2004] where different NoC topologies and each topology with different router and NA configuration is explored.

4.3. Traffic Characterization

The communication types expected in a NoC range across virtual wires, memory access, audio/video stream, interrupts, and others. Many combinations of topology, protocol, packet sizes, and flow control mechanisms exist for the efficient communication of one or more predominant traffic patterns. For example, in Kumar et al. [2002], packet-switched NoC concepts have been applied to a 2D mesh network topology, whereas in Guerrier and Greiner [2000], such concepts have been applied to a butterfly fat-tree topology. The design decisions were based on the traffic expected in the respective systems. Characterizing the expected traffic is an important first step towards making sound design decisions.

A NoC must accommodate different types of communication. We have realized that, regardless of the system composition, clustering, topology, and protocol, the traffic within a system will fall into one of three categories.

(1) *Latency Critical.* Latency critical traffic is traffic with stringent latency demands such as for critical interrupts, memory access, etc. These often have low payload.

(2) *Data Streams.* Data streaming traffic have high payload and demand QoS in terms of bandwidth. Most often it is large, mostly fixed bandwidth, which may be jitter critical. Examples are MPEG data, DMA access, etc.

(3) *Miscellaneous.* This is traffic with no specific requirements of commitment from the network.

This categorization is a guideline rather than a hard specification and is presented as a superset of possible traffic types. Bolotin et al. [2004] provide a more refined traffic categorization, combining the transactions at the network boundary with service requirements, namely, signaling, real-time, read/write (RD/WR), and block transfer. In relation to the previous categorization, signaling is latency critical, real-time is data streaming, and RD/WR and block transfer are both miscellaneous with the message size as the distinguishing factor. Though one or more of the traffic patterns may be predominant in the SoC, it is important to understand that a realistic NoC design should be optimized for a mix of traffic patterns. The conclusions of a case study of NoC routing mechanism for three traffic conditions with a fixed number of flits per packet as presented in Ost et al. [2005] can thus be enriched by using nonuniform packet size and relating them to the traffic categories presented.

It is important to understand the bandwidth requirements of the listed traffic types for a given application, and accordingly map the IP cores on the choosen NoC topology. Such a study is done in Murali and Micheli [2004a]. NMAP (now called SUNMAP [Murali and Micheli 2004b]), a fast mapping algorithm that minimizes the average communication delay with minimal path and split traffic routing in 2D mesh, is compared with greedy and partial branch-and-bound algorithms. It is shown to produce results of higher merit (reduced packet latency) for DSP benchmarks. Another dimension in the mapping task is that of allocating guaranteed communication resources. In Goossens et al. [2005] and Hansson et al. [2005] approaches to this task are explored for the ÆTHEREAL NoC.

Specific to the data stream type traffic described, Rixner et al. [1998] have identified unique qualities relating to the interdependencies between the media streams and frequency of such streams in the system. It is called the streaming programming model. The basic premises of such programming is static analysis of the application to optimize the mapping effort based on prior knowledge of the traffic pattern so as to minimize communication. The communication architecture tuner (CAT) proposed by Lahiri et al. [2000] is a hardware-based approach that does runtime analysis of traffic and manipulates the underlying NoC protocol. It does this by monitoring the internal state and communication transactions of each core and then predicts the relative importance of each communication event in terms of its potential impact on different system-level performance metrics such as number of deadline misses, average processing time, etc.

The various blocks of NoC can be tuned for optimum performance with regard to a specific traffic characteristic, or the aim can be more general, towards a one-fits-all network, for greater flexibility and versatility.

5. NETWORK ANALYSIS

The most interesting and universally applicable parameters of NoC are *latency*, *bandwidth*, *jitter*, *power consumption*, and *area usage*. Latency, bandwidth and jitter can be classified as performance parameters, while power consumption and area usage are the cost factors. In this section, we will discuss the analysis and presentation of results in relation to these parameters.

5.1. Performance Parameters and Benchmarks

Specifying a single one of the performance parameters previously introduced is not sufficient to confer a properly constrained NoC behavior. The following example illustrates this.

Given a network during normal operation, it is assumed that the network is not overloaded. For such a network, all data is guaranteed to reach its destination when

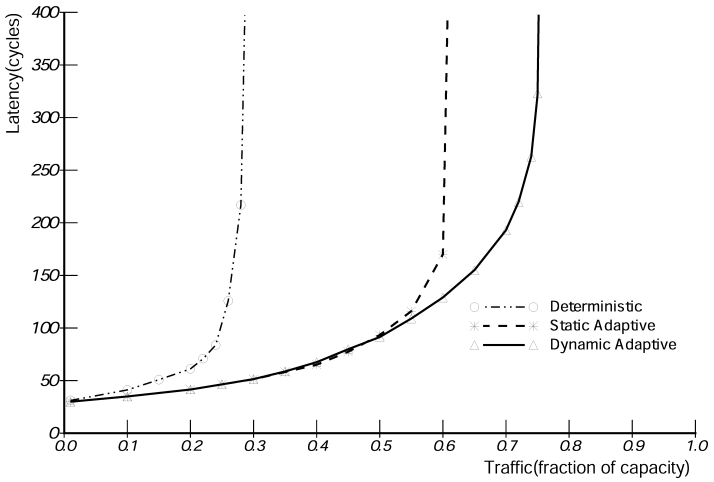

Fig. 18. Latency vs. network load for different routing schemes. The figure shows how the employment of more complex routing schemes move the point at which the network saturates (reprinted from Dally and Aoki [1993] Fig. 5, ©1993, with permission from IEEE).

employing a routing scheme in which no data is dropped (see Section 3.2.2, delay routing model). This means that, as long as the capacity of the network is not exceeded, any transmission is guaranteed to succeed (any required bandwidth is guaranteed). However, nothing is stated concerning the transmission latency which may well be very high in a network operated near full capacity. As shown in Figure 18, the exact meaning of which will be explained later, the latency of packets rise in an exponential manner as the *network load* increases. The exact nature of the network load will be detailed later in this section. It is obvious that such guarantees are not practically usable. We observe that the bandwidth specification is worthless without a bound on the latency as well. This might also be presented in terms of a maximum time window within which the specified bandwidth would always be reached, that is, the jitter of the data stream (the spread of the latencies). Jitter is often a more interesting parameter in relation to bandwidth than latency as it describes the temporal evenness of the data stream.

Likewise, a guaranteed bound on latency might be irrelevant if the bandwidth supported at this latency is insufficient. Thus latency, bandwidth, and jitter are closely related. Strictly speaking, one should not be specified without at least one of the others.

At a higher abstraction level, performance parameters used in evaluating multicomputer networks in general have been adopted by NoC researchers. These include *aggregated bandwidth*, *bisection bandwidth*, *link utilization*, *network load*, etc. The aggregate bandwidth is the accumulated bandwidth of all links, and the bisection bandwidth is the minimum collective bandwidth across links that, when cut, separate the network into two equal set of nodes. Link utilization is the load on the link compared with the total bandwidth available. The network load can be measured as a fraction of the network capacity, as normalized bandwidth. The network capacity is the maximum capacity of the network for a uniform traffic distribution, assuming that the most heavily loaded links are located in the network bisection. These and other aspects of network performance metrics are discussed in detail in Chapter 9 of Duato et al. [2003].

For highly complex systems, such as full-fledged computer systems including processor(s), memory, and peripherals, the individual parameters may say little about the

overall functionality and performance of the system. In such cases, it is customary to make use of benchmarks. NoC-based systems represents such complexity, and benchmarks would be natural to use in its evaluation. Presenting performance in the form of benchmark results would help clarify the effect of implemented features in terms of both performance benefits (latency, jitter, and bandwidth) and implementation and operation costs (area usage and power consumption). Benchmarks would thus provide a uniform plane of reference from which to evaluate different NoC architectures. At present, no benchmark system exists explicitly for NoC, but its development is an exciting prospect. In Vaidya et al. [2001], examples from the NAS benchmarks [Bailey et al. 1994] were used, in particular Class-A NAS-2. This is a set of benchmarks that has been developed for the performance evaluation of highly parallel supercomputers which mimic the computation and data movement characteristics of large scale computational fluid dynamics applications. It is questionable, however, how such parallel computer benchmarks can be used in NoC as the applications in SoCs are very different. In particular, SoC applications are generally highly heterogeneous, and the traffic patterns therein likewise. Another set of benchmarks, used as the basis of NoC evaluation in Hu and Marculescu [2004a], are the embedded system synthesis suite (E3S) [Dick 2002].

5.2. Presenting Results

Generally it is necessary to simplify the multidimensional performance space. One common approach is to adjust a single aspect of the design, while tracking the effect on the performance parameters. An example is tracking the latency of packets, while adjusting the bandwidth capabilities of a certain link within the network, or the amount of background traffic generated by the test environment. In Section 5.2.1, we will give specific examples of simple yet informative ways of communicating results of NoC performance measurements.

Since the NoC is a shared, segmented communication structure wherein many individual data transfer sessions can take place in parallel, the performance measurements there in, not only on the traffic being measured therein, but also on the other traffic in the network, the *background traffic*. The degree of background traffic is often indicated by the network load as described earlier. Though very simple, this definition makes valuable sense in considering a homogeneous, uniformly loaded network. One generally applicable practical method for performance evaluation is thus generating a uniform randomly-distributed background traffic so that the network load reaches a specified point. Test packets can then be sent from one node to another, according to the situation that one desires to investigate, and the latencies of these packets can be recorded (see example (i) in Section 5.2.1).

Evenly distributed traffic, however, may cloud important issues of the network performance. In Dally and Aoki [1993], the degree of symmetry of the traffic distribution in the network was used to illustrate aspects of different types of routing protocols, adaptive and deterministic. The adaptive protocol resulted in a significant improvement of throughput over the deterministic one for nonuniform traffic but had little effect on performance with uniformly distributed traffic. The reason for this is that the effect of adaptive protocols is to even out the load to avoid hotspots, thus making better use of the available network resources. If the bulk load is already evenly distributed, there is no advantage. Also traffic parameters, like number of packets and packet size, can have a great influence on performance, for example, in relation to queueing strategies in nodes.

There are many ways to approach the task of presenting test results. The performance space is a complex, multidimensional one, and there are many pitfalls to be avoided in order to display intelligible and valuable information about the performance of a

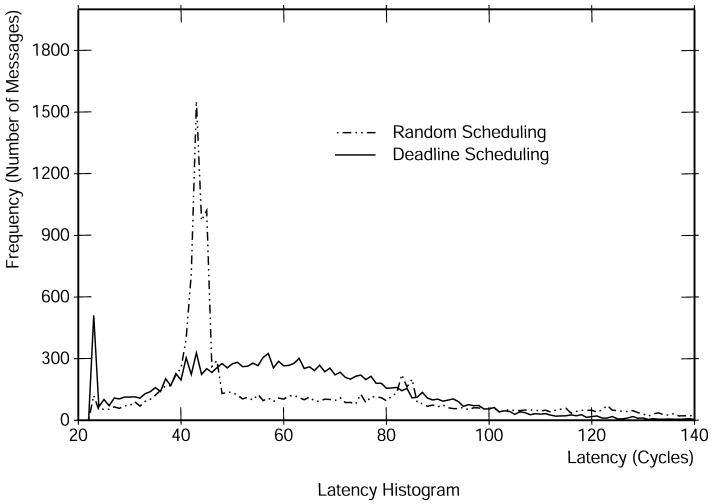

Fig. 19. Number of messages as a function of latency of message (latency distribution), for two scheduling schemes (reprinted from Dally [1992] Fig. 17, ©1992, with permission from IEEE).

network. Often the presented results fail to show the interesting aspects of the network. It is easy to get lost in the multitude of possible combinations of test parameters. This may lead to clouding (or at worst failure to properly communicate), the relevant aspects of the research. Though the basis for performance evaluation may vary greatly, it is important for researchers to be clear about the evaluation conditions, allowing others to readily and intuitively grasp the potential of a newly developed idea and the value of its usage in NoC.

5.2.1. Examples. We will now give some specific examples that we find clearly communicate the performance of the networks being analyzed. What makes these examples good are their simplicity in providing a clear picture of some very fundamental properties of the involved designs.

(i) *Average latency vs. network load.* In Dally and Aoki [1993], this is used to illustrate the effect of different routing schemes. Figure 18 is a figure from the article, showing how the average latency of the test data grows exponentially as the background traffic load of the network is increased. In the presented case, the *throughput saturation point*, the point at which the latency curve bends sharply upwards, is shifted right as more complex routing schemes are applied. This corresponds to a better utilization of available routing resources. The article does not address the question of cost factors of the implementation.

(ii) *Frequency of occurrence vs. latency of packet.* Displaying the average latency of packets in the network may work well for establishing a qualitative notion of network performance. Where more detail is needed, a histogram or similar graph showing the distribution of latencies across the delay spectrum is often used with great effect. This form of presentation is used in Dally [1992] to illustrate the effect of routing prioritization schemes on the latency distribution. Figure 19, taken from the article, shows the effect of *random scheduling* and *deadline scheduling*. Random scheduling schedules the packets for transmission in a random fashion, while deadline scheduling prioritize packets according to how long they have been waiting (oldest-packet-first). It is shown how the choice of scheduling affects the distribution of latencies of messages. In

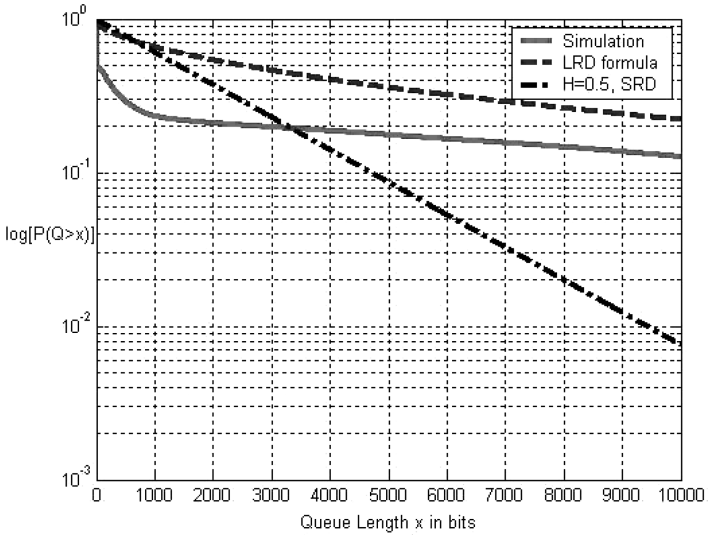

Fig. 20. The probability of queue length exceeding buffer size. The results for two models based on stochastic processes, LRD (Long Range Dependent) and SRD (Short Range Dependent), are plotted along with simulation results for comparison (reprinted from Varatkar and Marculescu [2002] Fig. 6).

Bjerregaard and Sparsø [2005b], such a latency distribution graph is also used to display how a scheduling scheme provides hard latency bounds in that the graph is completely empty beyond a certain latency.

(iii) *Jitter vs. network load.* The jitter of a sequence of packets is important when dimensioning buffers in the network nodes. High jitter (bursty traffic) needs large buffers to compensate in order to avoid congestion resulting in suboptimal utilization of routing resources. This issue is especially relevant in multimedia application systems with large continuous streams of data such as that presented in Varatkar and Marculescu [2002]. In this work, statistical mathematical methods are used to analyze the traffic distribution. Figure 20, taken from the article, explores the use of two different models based on stochastic processes for predicting the probability that the queue length needed to avoid congestion exceeds the actual buffer size in the given situation. The models displayed in the figure are LRD (Long Range Dependent) or *self-similar*, and SRD (Short Range Dependent) or *Markovian* stochastic processes. In the figure, these models are compared with simulation results. The contributions of the paper include showing that LRD processes can be used effectively to model the bursty traffic behavior at chip level, and the figure shows how indeed the predictions of the LRD model comes closer to the simulation results than those of the SRD model.

5.3. Cost Factors

The cost factors are basically power consumption and area usage. A comparative analysis of cost of NoC is difficult to make. As is the case for performance evaluation, no common ground for comparison exists. This would require different NoC being demonstrated for the same application which is most often not the case. Hence a somewhat broad discussion of cost in terms of area and power cost is presented in this section.

The power consumption of the communication structure in large single-chip systems is a major concern, especially for mobile applications. As discussed earlier, the power

used for global communication does not scale with technology scaling, leading to increased power use by communication relative to power use by processing. In calculating the power consumption of a system, there are two main terms: (i) power per communicated bit and (ii) idle power. Depending on the traffic characteristics in the network, different implementation styles will be more beneficial with regards to power usage. In Nielsen and Sparsø [2001], a power analysis of different low-power implementations of on-chip communication structures was made. The effects on power consumption of scaling a design were seen, and a bus design was compared with torus connected grid design (both synchronous and asynchronous implementations). Asynchronous implementation styles (discussed in Section 3.3.1), are beneficial for low network usage since they have very limited power consumption when idle, but use more power per communicated bit due to local control overhead. Technology scaling, however, leads to increased leakage current, resulting in an increasing static power use in transistors. Thus the benefit of low idle power in asynchronous circuits may dwindle.

From a system-level perspective, knowledge of network traffic can be used to control the power use of the cores. Interest has been expressed in investigating centralized versus distributed power management (DPM) schemes. Centralized power managers (PM) are a legacy in bus-based systems. Since NoC is most often characterized by distributed routing control, naturally distributed PMs, such as those proposed in Benini and Micheli [2001] and Simunic and Boyd [2002], would be useful. In both of these studies, conceptually there is a node-centric and network-centric PM. The node-centric PM controls the powering up or down of the core. The network-centric PM is used for overall load-balancing and to provide some estimations to the node-centric PM of incoming requests, thus masking the core's wake-up cost by precognition of traffic. This type of power management is expected to be favored to reduce power consumption in future NoCs. The results, presented in Simunic and Boyd [2002], show that, with only node PM, the power saving range from a factor of 1.5 to 3 compare to no power managers. Combining dynamic voltage scaling with DPM gives overall saving of a factor of 3.6. The combined implementation of node and network-centric management approaches shows energy savings of a factor of 4.1 with the performance penalty reduced by a minimum 15% compared to node-only PM. Unlike these dynamic runtime energy monitors, in Hu and Marculescu [2004b], a system-level energy-aware mapping and scheduling (EAS) algorithm is proposed which statically schedules both communication transactions and computation tasks. For experiments done on 2D mesh with minimal path routing, energy savings of 44% are reported when executing complex multimedia benchmarks.

A design constraint of NoC less applicable to traditional multicomputer networks lies in the area usage. A NoC is generally required to take up less than 5% of the total chip area. For a 0.13 μm SoC with one network node per core and an average core size of 2×2 mm (approximately 100 cores on a large chip), this corresponds to 0.2 mm^2 per node. One must also remember that the NA will use some area, depending on the complexity of the features that it provides. Trade-off decisions which are applicable to chip design in general and not particular to NoC are beyond the scope of this survey. At the network level, many researchers have concluded that buffering accounts for the major portion of the node area, hence wormhole routing has been a very popular choice in NoCs (see Section 3.2.2). As examples of an area issue related to global wires, introducing *fat wires*, that is, the usage of wide and tall top-level metal wires for global routing, may improve the power figures but at the expense of area [Sylvester and Keutzer 2000].

6. NOC EXAMPLES

In this section, we briefly recapitulate a handful of specific NoC examples, describing the design choices of actual implementations and the accompanying work by the research

groups behind them. This is by no means a complete compilation of existing NoCs, there are many more, rather the purpose of this section is to address a representative set: ÆTHEREAL, NOSTRUM, SPIN, CHAIN, MANGO, and XPIPES. In Moraes et al. [2004], a list in tabular form is provided which effectively characterizes many of the NoCs not covered in the following.

(1) **ÆTHEREAL.** The ÆTHEREAL, developed at Philips, is a NoC that provides guaranteed throughput (GT) alongside best-effort (BE) service [Rijpkema et al. 2001; Goossens et al. 2002; Wielage and Goossens 2002; Dielissen et al. 2003; Jantsch and Tenhunen 2003] (pgs: 61-82) [Rijpkema et al. 2003; Radulescu et al. 2004; Goossens et al. 2005]. In the ÆTHEREAL the guaranteed services pervade as a requirement for hardware design and also as a foundation for software programming. The router provides both GT and BE services. All routers in the network have a common sense of time, and the routers forward traffic based on slot allocation. Thus a sequence of slots implement a virtual circuit. GT traffic is connection-oriented, and in early router instantiations, did not have headers as the next hop was determined by a local slot table. In recent versions, the slot tables have been removed to save area, and the information is provided in a GT packet header. The allocation of slots can be setup statically, during an initialization phase, or dynamically, during runtime. BE traffic makes use of non-reserved slots and of any slots reserved but not used. BE packets are used to program the GT slots of the routers. With regard to buffering, input queuing is implemented using custom-made hardware fifos to keep the area costs down. The ÆTHEREAL connections support a number of different transaction types, such as read, write, acknowledged write, test and set, and flush, and, as such, it is similar to existing bus protocols. In addition, it offers a number of connection types including narrowcast, multicast, and simple.

In Dielissen et al. [2003], an ÆTHEREAL router with 6 bidirectional ports of 32 bits was synthesized in 0.13 μm CMOS technology. The router had custom-made BE input queues depth of 24 words per port. The total area was 0.175 mm^2, and the bandwidth was 500 MHz $\times$ 32 bits = 16 Gbit/s per port. A network adapter with 4 standard socket interfaces (either master or slave; OCP, DTL, or AXI based) was also reported with an area of 0.172 mm^2 implemented in the same technology.

In Goossens et al. [2005] and Pestana et al. [2004], an automated design flow for instantiation of application specific ÆTHEREAL is described. The flow uses XML to input various parameters such as traffic characteristics, GT and BE requirements, and topology. A case study of MPEG codec SoC is used to validate and verify the optimizations undertaken during the automated flow.

(2) **NOSTRUM.** The work of researchers at KTH in Stockholm has evolved from a system-level chip design approach [Kumar et al. 2002; Jantsch and Tenhunen 2003; Zimmer and Jantsch 2003; Millberg et al. 2004]. Their emphasis has been on architecture and platform-based design targeted towards multiple application domains. They have recognized the increasing complexity of working with high-density VLSI technologies and hence highlighted advantages of a grid-based, router-driven communication media for on-chip communication.

Also the implementation of guaranteed services has also been a focus point of this group. In the NOSTRUM NoC, guaranteed services are provided by so called *looped containers*. These are implemented by virtual circuits, using an explicit time division multiplexing mechanism which they call *Temporally Disjoint Networks* (TDN) (refer to Sections 3.2.2 and 3.2.3 for more details).

In Jantsch and Vitkowski [2005], the group addressed encoding issues and showed that lowering the voltage swing, then reestablishing reliability using error correction, actually resulted in better power saving than a number of dedicated power saving algorithms used for comparison.

(3) **SPIN.** The SPIN network (*Scalable Programmable Integrated Network*) [Guerrier and Greiner 2000; Andriahantenaina and Greiner 2003] implements a fat-tree topology with two one-way 32-bit datapaths at the link layer. The fat tree is an interesting choice of irregular network claimed in Leiserson [1985] to be "nearly the best routing network for a given amount of hardware." It is proven that, for any given amount of hardware, a fat tree can simulate any other network built from the same amount of hardware with only a polylogarithmic slowdown in latency. This is in contrast to, for example, two-dimensional arrays or simple trees which exhibit polynomial slowdown when simulating other networks and, as such, do not have any advantage over a sequential computer.

In SPIN, packets are sent via the network as a sequence of flits each of size 4 bytes. Wormhole routing is used with no limit on packet size. The first flit contains the header, with one byte reserved for addressing, and the last byte of the packet contains the payload checksum. There are three types of flits; first, data, and last. Link-level flow control is used to identify the flit type and act accordingly upon its content. The additional bytes in the header can be used for packet tagging for special services and for special routing options. The performance of the network was evaluated primarily based on uniform randomly distributed load (see Section 5). It was noted that random hick-ups can be expected under high load. It was found that the protocol accounts for about 31% of the total throughput, a relatively large overhead. In 2003, a 32-port SPIN network was implemented in a 0.13 μm CMOS process, the total area was 4.6 mm^2 (0.144 mm^2 per port), for an accumulated bandwidth of about 100 Gbits/s.

(4) **CHAIN.** The CHAIN network (*CHip Area INterconnect*) [Bainbridge and Furber 2002], developed at the University of Manchester, is interesting in that it is implemented entirely using asynchronous, or clockless, circuit techniques. It makes use of delay insensitive 1-of-4 encoding, and source routes BE packets. An easy adaption along a path consisting of links of different bit widths is supported. CHAIN is targeted for heterogeneous low-power systems in which the network is system specific. It has been implemented in a smart card which benefits from the low idle power capabilities of asynchronous circuits. Work from the group involved with CHAIN concerns prioritization in asynchronous networks. In Felicijan et al. [2003], an asynchronous low-latency arbiter was presented, and its use in providing differentiated communication services in SoC was discussed, and in Felicijan and Furber [2004], a router implementing the scheme was described.

(5) **MANGO.** The MANGO network (*Message-passing Asynchronous Network-on-chip providing Guaranteed services over OCP interfaces*), developed at the Technical University of Denmark, is another clockless NoC, targeted for coarse-grained GALS-type SoC [Bjerregaard 2005]. MANGO provides connectionless BE routing as well as connection-oriented guaranteed services (GS) [Bjerregaard and Sparsø 2005a]. In order to make for a simple design, the routers implement virtual channels (VCs) as separate physical buffers. GS connections are established by allocating a sequence of VCs through the network. While the routers themselves are implemented using area efficient bundled-data circuits, the links implement delay insensitive signal encoding. This makes global timing robust because no timing assumptions are necessary between routers. A scheduling scheme called ALG (*Asynchronous Latency*

Guarantees) [Bjerregaard and Sparsø 2005b] schedules access to the links, allowing latency guarantees to be made which are not inversely dependent on the bandwidth guarantees as is the case in TDM-based scheduling schemes. Network adapters provide OCP-based standard socket interfaces based on the primitive routing services of the network [Bjerregaard et al. 2005]. This includes support for interrupts based on virtual wires. The adapters also synchronize the clocked OCP interfaces to the clockless network.

(6) **XPIPES**. xpipes [Osso et al. 2003] and the accompanying NetChip compiler (a combination of xpipesCompiler [Jalabert et al. 2004] and SUNMAP [Murali and Micheli 2004b]) were developed by the University of Bologna and Stanford University. xpipes consists of soft macros of switches and links that can be turned into instance-specific network components at instantiation time. It promotes the idea of pipelined links with a flexible number of stages to increase throughput. A go-back-N retransmission strategy is implemented as part of link-level error control which reduces switch complexity, though at considerable delay since each flit is not acknowledged until it has been transmitted across the destination switch. The error is indicated by a CRC block running concurrently with switch operation. Thus the xpipes architecture lends itself to be robust to interconnect errors. Overall, delay for a flit to traverse from across one link and node is 2N + M cycles, where N is number of pipeline stages and M the switch stages. The xpipesCompiler is a tool to automatically instantiate an application-specific custom communication infrastructure using xpipes components. It can tune flit size, degree of redundancy of the CRC error-detection, address space of cores, number of bits used for packet sequence count, maximum number of hops between any two network nodes, number of flit size, etc.

In a top-down design methodology, once the SoC floorplan is decided, the required network architecture is fed into the xpipesCompiler. Examples of compiler optimization include removing redundant buffers from missing output ports of switches, sharing signals common to objects, etc. Via case studies presented in Bertozzi et al. [2005], the NetChip compiler has been validated for mesh, torus, hypercube, Clos, and butterfly NoC topologies for four video processing applications. Four routing algorithms, dimension-ordered, minimum-path, traffic splitting across minimum-path, and traffic splitting across all paths, is also part of the case study experiments. The floorplan of switches and links of NoC takes the IP block size into consideration. Results are available for average hop delay, area and power for mapping of each of the video application on the topologies. A lightweight implementation, named xpipes-lite, presented in Stergiou et al. [2005], is similar in to xpipes in concept, but is optimized for link latency, area and power, and provides direct synthesis path from SystemC description.

7. SUMMARY

NoC encompasses a wide spectrum of research, ranging from highly abstract software related issues, across system topology to physical level implementation. In this survey, we have given an overview of activities in the field. We have first stated the motivation for NoC and given an introduction of the basic concepts. In order to avoid the wide range of topics relevant to large scale IC design in general, we have assumed a view of NoC as a subset of SoC.

From a system-level perspective, NoC is motivated by the demand for a well structured design approach in large scale SoCs. A modularized design methodology is needed in order to make efficient use of the ever increasing availability of on-chip resources

in terms of the number of transistors and wiring layers. Likewise, programming these systems necessitates clear programming models and predictable behavior. NoC has the potential to provide modularity through the use of standard sockets such as OCP and predictability through the implementation of guaranteed communication services. From a physical-level perspective, with scaling of technologies into the DSM region, the increasing impact of wires on performance forces a differentiation between local and global communication. In order for global communication structures to exhibit scalability and high performance, segmentation, wire sharing, and distributed control is required.

In structuring our work, we have adopted a layered approach similar to OSI and divided NoC research into four areas: system, network adapter, network and link research. In accordance with the view of NoC as a subset of SoC, we have dealt first with the latter three areas of research which relate directly to the NoC implementation. Thereafter, we have focused on system-level aspects.

The network adapter orthogonalizes communication and computation, enabling communication-centric design. It is thus the entity which enables a modularized design approach. Its main task is to decouple the core from the network with the purpose of providing high-level network-agnostic communication services based on the low-level routing primitives provided by the network hardware. In implementing standard sockets, IP reuse becomes feasible, and the network adapter may, therefore, hold the key to the commercial success of NoC.

At the network level, issues such as network topology, routing protocols, flow control, and quality-of-service are dominant. With regards to topology, NoC is restricted by a 2D layout. This has made grid-type topologies a widespread choice. We have reviewed the most common routing schemes, store-and-forward, wormhole and virtual cut-through routing, and concluded that wormhole routing is by far the most common choice for NoC designs. The use of virtual channels in avoiding deadlocks and providing guaranteed services was illustrated, and the motivation for guaranteed services was discussed. The predictability that such services incur facilitates easy system integration and analytical system verification, particularly relevant for real-time systems.

Unlike macronetworks, in NoC, the network adapter and network functionality is often implemented in hardware rather than in software. This is so because NoC-based systems are more tightly bound, and simple, fast, power-efficient solutions are required.

Link-level research is much more hardware oriented. We have covered topics like synchronization, that is, between clock domains, segmentation, and pipelining of links, in order to increase bandwidth and counteract the physical limitations of DSM technologies, on-chip signaling such as low-swing drivers used to decrease the power usage in links, and future technologies such as on-chip wave guides, and optical interconnects. We have also discussed the reliability of long links, which are susceptible to a number of noise sources: crosstalk, ground bounce, EMI and intersymbol interference. Segmentation helps keep the effect of these at bay since the shorter a wire is, the less influence they will have. Error detection and correction in on-chip interconnects was discussed, but this is not a dominating area of research. Different encoding schemes were discussed in relation to increasing bandwidth as well as reducing power consumption.

NoC facilitates communication-centric design as opposed to traditional computation-centric design. From a system-level perspective, we have addressed topics relating to the role of NoC in SoC design flows. Key issues are modeling, design methodology, and traffic characterization. The purpose of modeling is to evaluate trade-offs with regard to global traffic in terms of power, area, design time, etc., while adhering to application requirements. With regard to design methodology, we identify two important characteristics of NoC, by which we classify a number of existing NoC solutions:

(i) parametrizability of the NoC as a system level block and (ii) granularity of the NoC components by which the NoC is assembled. These characteristics greatly influence the nature of the design flow enabled by the particular NoC. As a tool for identifying general requirements of a NoC, we have identified a set of traffic types, latency-critical, data-streams and miscellaneous traffic, which span the spectrum of possible traffic in a NoC-based system.

The basic performance parameters of NoC are latency, bandwidth and jitter. The basic cost factors are power consumption and area usage. At a higher level of abstraction, terms like aggregate bandwidth, bisection bandwidth, link utilization and network load can be used. These originate in multicomputer network theory and relate to data movement in general. Stepping up yet another abstraction level, benchmarks can be used for performance analysis. Currently no benchmarks exist specifically for NoC, but the use of benchmarks for parallel computers, as well as embedded systems benchmarks, has been reported.

Six case studies are conducted, explaining the design choices of the ÆTHEREAL, NOSTRUM, SPIN, CHAIN, MANGO and XPIPES NoC implementations. CHAIN and XPIPES target a platform-based design methodology in which a heterogeneous network can be instantiated for a particular application. ÆTHEREAL, NOSTRUM, and MANGO implement more complex features such as guaranteed services, and target a methodology which draws closer to backbone-based design. SPIN differs from the others in that it implements a fat tree rather than a grid-type topology. CHAIN and MANGO also differ in that they are implemented entirely using clockless circuit techniques and, as such, inherently support globally asynchronous and locally synchronous (GALS) systems.

Continued technology scaling enables large scale SoC. NoCs facilitate a modular, scalable design approach that overcomes both system and physical-level issues. The main job of the NoC designer of the future will be to dimension and structure the network according to the communication needs of the SoC. At present, an interesting challenge lies in specifying ways to define these needs.

ACKNOWLEDGMENTS

We would like to thank professors Jens Sparsø and Jan Madsen of the Department for Informatics and Mathematical Modelling (IMM) at the Technical University of Denmark (DTU) for their tireless effort in helping us review, iterate, and structure this survey. Also our grateful thanks to professor Axel Jantsch (KTH—Stockholm, Sweden) and Andrei Radulescu (Phillips—Eindhoven, Netherlands) for their valuable review of the survey as it was closing in on its final form, and to Mihai Budiu (Carnegie Mellon University—Pittsburgh, USA) for comments and suggestions. Finally, the extensive comments of the anonymous reviewers have helped in taking the survey to its final form.

REFERENCES

AGARWAL, A. 1999. The Oxygen project—Raw computation. *Scientific American*, 44–47.

AGGARWAL, A. AND FRANKLIN, M. 2002. Hierarchical interconnects for on-chip clustering. In *Proceedings of the 16th International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE Computer Society, 602–609.

AHONEN, T., SIGENZA-TORTOSA, D. A., BIN, H., AND NURMI, J. 2004. Topology optimization for application-specific networks-on-chip. In *International Workshop on System Level Interconnect Prediction (SLIP)*. ACM, 53–60.

AL-TAWIL, K. M., ABD-EL-BARR, M., AND ASHRAF, F. 1997. A survey and comparison of wormhole routing techniques in a mesh networks. *IEEE Network 11*, 38–45.

AMDE, M., FELICIJAN, T., EDWARDS, A. E. D., AND LAVAGNO, L. 2005. Asynchronous on-chip networks. *IEE Proceedings of Computers and Digital Techniques 152*, 273–283.

ANDREASSON, D. AND KUMAR, S. 2004. On improving best-effort throughput by better utilization of guaranteed-throughput channels in an on-chip communication system. In *Proceeding of 22th IEEE Norchip Conference*.

ANDREASSON, D. AND KUMAR, S. 2005. Slack-time aware routing in NoC systems. In *International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2353–2356.

ANDRIAHANTENAINA, A. AND GREINER, A. 2003. Micro-network for SoC: Implementation of a 32-port spin network. In *Proceedings of Design, Automation and Test in Europe Conference and Exhibition*. IEEE, 1128–1129.

ARM. 2004. AMBA Advanced eXtensible Interface (AXI) Protocol Specification, Version 1.0. http://www.arm.com.

ARTERIS. 2005. A comparison of network-on-chip and busses. White paper. http://www.arteris.com/noc_whitepaper.pdf.

BAILEY, D., BARSZCZ, E., BARTON, J., BROWNING, D., CARTER, R., DAGUM, L., FATOOHI, R., FINEBERG, S., FREDERICKSON, P., LASINSKI, T., SCHREIBER, R., SIMON, H., VENKATAKRISHNAN, V., AND WEERATUNGA, S. 1994. RNR Tech. rep. RNR-94-007. NASA Ames Research Center.

BAINBRIDGE, J. AND FURBER, S. 2002. CHAIN: A delay-insensitive chip area interconnect. *IEEE Micro 22*, 5 (Oct.) 16–23.

BAINBRIDGE, W. AND FURBER, S. 2001. Delay insensitive system-on-chip interconnect using 1-of-4 data encoding. In *Proceedings of the 7th International Symposium on Asynchronous Circuits and Systems (ASYNC)*. 118–126.

BANERJEE, N., VELLANKI, P., AND CHATHA, K. S. 2004. A power and performance model for network-on-chip architectures. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE)*. IEEE, 1250–1255.

BEIGNE, E., CLERMIDY, F., VIVET, P., CLOUARD, A., AND RENAUDIN, M. 2005. An asynchronous NOC architecture providing low latency service and its multi-level design framework. In *Proceedings of the 11th International Symposium on Asynchronous Circuits and Systems (ASYNC)*. IEEE, 54–63.

BENINI, L. AND MICHELI, G. D. 2001. Powering network-on-chips. In *The 14th International Symposium on System Synthesis (ISSS)*. IEEE, 33–38.

BENINI, L. AND MICHELI, G. D. 2002. Networks on chips: A new SoC paradigm. *IEEE Comput. 35*, 1 (Jan.), 70–78.

BERTOZZI, D., JALABERT, A., MURALI, S., TAMHANKAR, R., STERGIOU, S., BENINI, L., AND DE MICHELI, G. 2005. NoC synthesis flow for customized domain specific multiprocessor Systems-on-Chip. In *IEEE Trans. Parall. Distrib. Syst.* 113–129.

BHOJWANI, P. AND MAHAPATRA, R. 2003. Interfacing cores with on-chip packet-switched networks. In *Proceedings of the 16th International Conference on VLSI Design*. 382–387.

BJERREGAARD, T. 2005. The MANGO clockless network-on-chip: Concepts and implementation. Ph.D. thesis, Informatics and Mathematical Modeling, Technical University of Denmark, Lyngby, Denmark.

BJERREGAARD, T., MAHADEVAN, S., OLSEN, R. G., AND SPARSØ, J. 2005. An OCP compliant network adapter for gals-based soc design using the MANGO network-on-chip. In *Proceedings of International Symposium on System-on-Chip (ISSoC)*. IEEE.

BJERREGAARD, T., MAHADEVAN, S., AND SPARSØ, J. 2004. A channel library for asynchronous circuit design supporting mixed-mode modeling. In *Proceedings of the 14th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*. Springer, 301–310.

BJERREGAARD, T. AND SPARSØ, J. 2005a. A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE)*. IEEE, 1226–1231.

BJERREGAARD, T. AND SPARSØ, J. 2005b. A scheduling discipline for latency and bandwidth guarantees in asynchronous network-on-chip. In *Proceedings of the 11th International Symposium on Advanced Resear. in Asynchronous Circuits and Systems*. IEEE, 34–43.

BOGLIOLO, A. 2001. Encodings for high-performance energy-efficient signaling. In *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED)*. 170–175.

BOLOTIN, E., CIDON, I., GINOSAUR, R., AND KOLODNY, A. 2004. QNoC: QoS architecture and design process for network-on-chip. *J. Syst. Archit. 50*, 2-3, 105–128.

CATTHOOR, F., CUOMO, A., MARTIN, G., GROENEVELD, P., RUDY, L., MAEX, K., DE STEEG, P. V., AND WILSON, R. 2004. How can system level design solve the interconnect technology scaling problem. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE)*. IEEE, 332–337.

CHAPIRO, D. 1984. Globally-asynchronous locally-synchronous systems. Ph.D. thesis (Report No. STAN-CS-84-1026) Stanford University.

CHELCEA, T. AND NOWICK, S. M. 2001. Robust interfaces for mixed-timing systems with application to latency-insensitive protocols. In *Proceedings of the 38th Design Automation Conference (DAC)*. IEEE, 21–26.

CHIU, G.-M. 2000. The odd-even turn model for adaptive routing. *IEEE Trans. Parall. Distrib. Syst. 11*, 729–738.

COLE, R. J., MAGGS, B. M., AND SITARAMAN, R. K. 2001. On the benefit of supporting virtual channels in wormhole routers. *J. Comput. Syst. Sciences 62*, 152–177.

CULLER, D. E., SINGH, J. P., AND GUPTA, A. 1998. *Parallel Computer Architecture: A Hardware/Software Approach*. 1st Ed. Morgan Kaufmann.

DALLY, W. J. 1990. Performance analysis of k-ary n-cube interconnection networks. *IEEE Trans. Comput. 39*, 6 (June) 775–785.

DALLY, W. J. 1992. Virtual-channel flow control. *IEEE Trans. Parall. Distrib. Syst. 3*, 2 (March) 194–205.

DALLY, W. J. AND AOKI, H. 1993. Deadlock-free adaptive routing in multicomputer networks using virtual channels. *IEEE Trans. Parall. Distrib. Syst. 4*, 4 (April) 466–475.

DALLY, W. J. AND SEITZ, C. L. 1987. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Trans. Comput. 36*, 5 (May) 547–553.

DALLY, W. J. AND TOWLES, B. 2001. Route packets, not wires: On-chip interconnection networks. In *Proceedings of the 38th Design Automation Conference (DAC)*. IEEE, 684–689.

DE MELLO, A. V., OST, L. C., MORAES, F. G., AND CALAZANS, N. L. V. 2004. Evaluation of routing algorithms on mesh based nocs. Tech. rep., Faculdade de Informatica PUCRS—Brazil.

DICK, R. 2002. Embedded system synthesis benchmarks suite. http://www.ece. northwestern.edu/ dickrp/e3s/.

DIELISSEN, J., RADULESCU, A., GOOSSENS, K., AND RIJPKEMA, E. 2003. Concepts and implementation of the phillips network-on-chip. In *Proceedings of the IP based SOC (IPSOC)*. IFIP.

DOBBELAERE, I., HOROWITZ, M., AND GAMAL, A. E. 1995. Regenerative feedback repeaters for programmable interconnections. *IEEE J. Solid-State Circuits 30*, 11 (Nov.) 1246–1253.

DOBKIN, R., GINOSAUR, R., AND SOTIRIOU, C. P. 2004. Data synchronization issues in GALS SoCs. In *Proceedings of the 10th IEEE International Symposium on Asynchronous Circuits and Systems*. IEEE, 170–179.

DUATO, J. 1993. A new theory of deadlock-free adaptive routing in wormhole networks. *IEEE Trans. Parall. Distrib. Syst. 4*, 12 (Dec.) 1320–1331.

DUATO, J. 1995. A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks. *IEEE Trans. Parall. Distrib. Syst. 6*, 10 (Oct.) 1055–1067.

DUATO, J. 1996. A necessary and sufficient condition for deadlock-free routing in cut-through and store-and-forward networks. *IEEE Trans. Parall. Distrib. Syst. 7*, 8 (Aug.) 841–854.

DUATO, J. AND PINKSTON, T. M. 2001. A general theory for deadlock-free adaptive routing using a mixed set of resources. *IEEE Trans. Parall. Distrib. Syst. 12*, 12 (Dec.) 1219–1235.

DUATO, J., YALAMANCHILI, S., AND NI, L. 2003. *Interconnection Networks: An Engineering Approach*. Morgan Kaufmann.

FELICIJAN, T., BAINBRIDGE, J., AND FURBER, S. 2003. An asynchronous low latency arbiter for quality of service (QoS) applications. In *Proceedings of the 15th International Conference on Microelectronics (ICM)*. IEEE, 123–126.

FELICIJAN, T. AND FURBER, S. B. 2004. An asynchronous on-chip network router with quality-of-service (QoS) support. In *Proceedings IEEE International SOC Conference*. IEEE, 274–277.

FITZPATRICK, T. 2004. System verilog for VHDL users. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE)*. IEEE Computer Society, 21334.

FORSELL, M. 2002. A scalable high-performance computing solution for networks on chips. *IEEE Micro 22*, 5, 46–55.

GAUGHAN, P. T., DAO, B. V., YALAMANCHILI, S., AND SCHIMMEL, D. E. 1996. Distributed, deadlock-free routing in faulty, pipelined, direct interconnection networks. *IEEE Trans. Comput. 45*, 6 (June) 651–665.

GENKO, N., ATIENZA, D., DE MICHELI, G., BENINI, L., MENDIAS, J., HERMIDA, R., AND CATTHOOR, F. 2005. A novel approach for network on chip emulation. In *International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2365–2368.

GERSTLAUER, A. 2003. Communication abstractions for system-level design and synthesis. Tech. Rep. TR-03-30, Center for Embedded Computer Systems, University of California, Irvine, CA.

GINOSAUR, R. 2003. Fourteen ways to fool your synchrononizer. In *Proceedings of the 9th International Symposium on Asynchronous Circuits and Systems*. IEEE, 89–96.

GLASS, C. J. AND NI, L. M. 1994. The turn model for adaptive routing. *J. ACM 41*, 874–902.

GOOSSENS, K., DIELISSEN, J., GANGWAL, O. P., PESTANA, S. G., RADULESCU, A., AND RIJPKEMA, E. 2005. A design flow for application-specific networks on chip with guaranteed performance to accelerate SOC design and verification. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE)*. IEEE, 1182–1187.

GOOSSENS, K., DIELISSEN, J., AND RADULESCU, A. 2005. Æthereal network on chip: Concepts, architectures and implementations. *IEEE Design Test Comput. 22*, 5, 414–421.

GOOSSENS, K., MEERBERGEN, J. V., PEETERS, A., AND WIELAGE, P. 2002. Networks on silicon: Combining best-effort and guaranteed services. In *Proceedings of the Design, Automation and Test in Europe Conference (DATE)*. IEEE, 196–200.

GUERRIER, P. AND GREINER, A. 2000. A generic architecture for on-chip packet-switched interconnections. In *Proceedings of the Design Automation and Test in Europe (DATE)*. IEEE, 250–256.

GUO, M., NAKATA, I., AND YAMASHITA, Y. 2000. Contention-free communication scheduling for array redistribution. *Parall. Comput. 26*, 1325–1343.

HANSSON, A., GOOSSENS, K., AND RADULESCU, A. 2005. A unified approach to constrained mapping and routing on networks-on-chip architectures. In *CODES/ISSS*. ACM/IEEE, 75–80.

HARMANCI, M., ESCUDERO, N., LEBLEBICI, Y., AND IENNE, P. 2005. Quantitative modeling and comparison of communication schemes to guarantee quality-of-service in networks-on-chip. In *International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1782–1785.

HAUCK, S. 1995. Asynchronous design methodologies: an overview. *Proceedings of the IEEE 83*, 1 (Jan.) 69–93.

HAVEMANN, R. H. AND HUTCHBY, J. A. 2001. High-performance interconnects: An integration overview. *Proceedings of the IEEE 89*, 5 (May) 586–601.

HAVERINEN, A., LECLERCQ, M., WEYRICH, N., AND WINGARD, D. 2002. SystemC based SoC communication modeling for the OCP protocol. White paper. http://www.ocpip.org.

HEILIGER, H.-M., NAGEL, M., ROSKOS, H. G., AND KURZ, H. 1997. Thin-film microstrip lines for mm and submm-wave on-chip interconnects. In *IEEE MTT-S Int. Microwave Symp. Digest*. Vol. 2. 421–424.

HO, R., MAI, K., AND HOROWITZ, M. 2003. Efficient on-chip global interconnects. In *Symposium on VLSI Circuits. Digest of Technical Papers.* IEEE, 271–274.

HO, R., MAI, K. W., AND HOROWITZ, M. A. 2001. The future of wires. *Proceedings of the IEEE 89*, 4 (April) 490–504.

HU, J. AND MARCULESCU, R. 2004a. Application-specific buffer space allocation for networks-on-chip router design. In *ICCAD*. IEEE/ACM, 354–361.

HU, J. AND MARCULESCU, R. 2004b. Energy-aware communication and task scheduling for network-on-chip architectures under real-time constraints. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE)*. IEEE, 10234–10240.

ITRS. 2001. International technology roadmap for semiconductors. Tech. rep., International Technology Roadmap for Semiconductors.

ITRS. 2003. International technology roadmap for semiconductors. Tech. rep., International Technology Roadmap for Semiconductors.

JALABERT, A., MURALI, S., BENINI, L., AND MICHELI, G. D. 2004. xpipesCompiler: A tool for instantiating application specific networks-on-chip. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE)*. IEEE, 884–889.

JANTSCH, A. 2003. Communication performance in networks-on-chip. http://www.ele. kth.se/ axel/presentations/2003/Stringent.pdf.

JANTSCH, A. AND TENHUNEN, H. 2003. *Networks on Chip*. Kluwer Academic Publishers.

JANTSCH, A. AND VITKOWSKI, R. L. A. 2005. Power analysis of link level and end-to-end data protection in networks-on-chip. In *International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1770–1773.

JUURLINK, B. H. H. AND WIJSHOFF, H. A. G. 1998. A quantitative comparison of parrallel computation models. *ACM Trans. Comput. Syst. 16*, 3 (Aug.) 271–318.

KAPUR, P. AND SARASWAT, K. C. 2003. Optical interconnects for future high performance intergrated circuits. *Physica E 16,* 3–4, 620–627.

KARIM, F., NGUYEN, A., AND DEY, S. 2002. An interconnect architecture for networking systems on chips. *IEEE Micro 22*, 36–45.

KARIM, F., NGUYEN, A., DEY, S., AND RAO, R. 2001. On-chip communication architecture for OC-768 network processors. In *Proceedings of the 38th Design Automation Conference (DAC)*. ACM, 678–683.

KIM, D., LEE, K., JOONG LEE, S., AND YOO, H.-J. 2005. A reconfigurable crossbar switch with adaptive bandwidth control for networks-on-chip. In *International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2369–2372.

KIM, K., LEE, S.-J., LEE, K., AND YOO, H.-J. 2005. An arbitration look-ahead scheme for reducing end-to-end latency in networks-on-chip. In *International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2357–2360.

KUMAR, S., JANTSCH, A., SOININEN, J.-P., FORSELL, M., MILLBERG, M., OBERG, J., TIENSYRJÄ, K., AND HEMANI, A. 2002. A network-on-chip architecture and design methodology. In *Proceedings of the Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE Computer Society, 117–124.

KURD, N., BARKATULLAH, J., DIZON, R., FLETCHER, T., AND MADLAND, P. 2001. Multi-GHz clocking scheme for Intel pentium 4 microprocessor. In *Digest of Technical Papers. International Solid-State Circuits Conference (ISSCC)*. IEEE, 404–405.

LAHIRI, K., RAGHUNATHAN, A., AND DEY, S. 2001. Evaluation of the traffic-performance characteristics of system-on-chip communication architectures. In *Proceedings of the 14th International Conference on VLSI Design*. IEEE, 29–35.

LAHIRI, K., RAGHUNATHAN, A., LAKSHMINARAYANA, G., AND DEY, S. 2000. Communication architecture tuners: A methodology for the design of high-performance communication architectures for system-on-chips. In *Proceedings of the Design Automation Conference, DAC*. IEEE, 513–518.

LEE, K. 1998. On-chip interconnects—gigahertz and beyond. *Solid State Technol. 41*, 9 (Sept.) 85–89.

LEISERSON, C. E. 1985. Fat-trees: Universal networks for hardware-efficient supercomputing. *IEEE Trans. Comput. c-34*, 10, 892–901.

LEROY, A., MARCHAL, P., SHICKOVA, A., CATTHOOR, F., ROBERT, F., AND VERKEST, D. 2005. Spatial division multiplexing: a novel approach for guaranteed throughput on nocs. In *CODES/ISSS*. ACM/IEEE, 81–86.

LIANG, J., LAFFELY, A., SRINIVASAN, S., AND TESSIER, R. 2004. An architecture and compiler for scalable on-chip communication. *IEEE Trans. VLSI Syst. 12*, 7, 711–726.

LIANG, J., SWAMINATHAN, S., AND TESSIER, R. 2000. ASOC: A scalable, single-chip communications architecture. In *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques*. 37–46.

LIU, J., ZHENG, L.-R., AND TENHUNEN, H. 2004. Interconnect intellectual property for network-on-chip (NoC). *J. Syst. Archite. 50*, 65–79.

LOGHI, M., ANGIOLINI, F., BERTOZZI, D., BENINI, L., AND ZAFALON, R. 2004. Analyzing on-chip communication in a MPSoC environment. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE)*. IEEE, 752–757.

MADSEN, J., MAHADEVAN, S., VIRK, K., AND GONZALEZ, M. 2003. Network-on-chip modeling for system-level multiprocessor simulation. In *Proceedings of the 24th IEEE International Real-Time Systems Symposium (RTSS)*. IEEE, 82–92.

MAHADEVAN, S., STORGAARD, M., MADSEN, J., AND VIRK, K. 2005. ARTS: A system-level framework for modeling MPSoC components and analysis of their causality. In *The 13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE Computer Society.

MAI, K., PAASKE, T., JAYASENA, N., HO, R., DALLY, W. J., AND HOROWITZ, M. 2000. Smart memories: A modular reconfigurable architecture. In *Proceedings of 27th International Symposium on Computer Architecture*. 161–171.

MEINCKE, T., HEMANI, A., KUMAR, S., ELLERVEE, P., OBERG, J., OLSSON, T., NILSSON, P., LINDQVIST, D., AND TENHUNEN, H. 1999. Globally asynchronous locally synchronous architecture for large high-performance ASICs. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*. Vol. 2. 512–515.

MILLBERG, M., NILSSON, E., THID, R., AND JANTSCH, A. 2004. Guaranteed bandwidth using looped containers in temporally disjoint networks within the nostrum network-on-chip. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE)*. IEEE, 890–895.

MIZUNO, M., DALLY, W. J., AND ONISHI, H. 2001. Elastic interconnects: Repeater-inserted long wiring capable of compressing and decompressign data. In *Proceedings of the International Solid-State Circuits Conference*. IEEE, 346–347, 464.

MORAES, F., CALAZANS, N., MELLO, A., MÖLLER, L., AND OST, L. 2004. HERMES: An infrastructure for low area overhead packet-switching networks on chip. *The VLSI Integration 38*, 69–93.

MULLINS, R. AND MOORE, A. W. S. 2004. Low-latency virtual-channel routers for on-chip networks. In *Proceedings of the 31st Annual International Symposium on Computer Architecture*. IEEE, 188–197.

MURALI, S. AND MICHELI, G. D. 2004a. Bandwidth-constrained mapping of cores onto noc architectures. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE)*. IEEE, 20896–20902.

MURALI, S. AND MICHELI, G. D. 2004b. SUNMAP: A tool for automatic topology selection and generation for NoCs. In *In Proceedings of the 41st Design Automation Conference (DAC)*. IEEE, 914–919.

MUTTERSBACH, J., VILLIGER, T., AND FICHTNER, W. 2000. Practical design of globally-asynchronous locally-synchronous systems. In *Proceedings of the 6th International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*. IEEE Computer Society, 52–59.

NAKAMURA, K. AND HOROWITZ, M. A. 1996. A 50% noise reduction interface using low-weight coding. In *Symposium on VLSI Circuits Digest of Technical Papers*. IEEE, 144–145.

NEDOVIC, N., OKLOBDZIJA, V. G., AND WALKER, W. W. 2003. A clock skew absorbing flip-flop. In *Proceedings of the International Solid-State Circuits Conference*. IEEE, 342–497.

NEEB, C., THUL, M., WEHN, N., NEEB, C., THUL, M., AND WEHN, N. 2005. Network-on-chip-centric approach to interleaving in high throughput channel decoders. In *International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1766–1769.

NIELSEN, S. F. AND SPARSØ, J. 2001. Analysis of low-power SoC interconnection networks. In *Proceedings of Nordchip 2001*. 77–86.

OBERG, J. 2003. *Clocking Strategies for Networks-on-Chip*. Kluwer Academic Publishers, 153–172.

OCPIP. 2003a. The importance of sockets in SoC design. White paper. http://www.ocpip.org.

OCPIP. 2003b. Open Core Protocol (OCP) Specification, Release 2.0. http://www.ocpip.org.

OKLOBDZIJA, V. G. AND SPARSØ, J. 2002. Future directions in clocking multi-GHz systems. In *Proceedings of the 2002 International Symposium on Low Power Electronics and Design, 2002 (ISLPED '02)*. ACM, 219.

OSSO, M. D., BICCARI, G., GIOVANNINI, L., BERTOZZI, D., AND BENINI, L. 2003. Xpipes: a latency insensitive parameterized network-on-chip architecture for multi-processor SoCs. In *Proceedings of 21st International Conference on Computer Design (ICCD)*. IEEE Computer Society, 536–539.

OST, L., MELLO, A., PALMA, J., MORAES, F., AND CALAZANS, N. 2005. MAIA—a framework for networks on chip generation and verification. In *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE.

PANDE, P., GRECU, C., JONES, M., IVANOV, A., AND SALEH, R. 2005. Effect of traffic localization on energy dissipation in NoC-based interconnect. In *International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1774–1777.

PANDE, P. P., GRECU, C., IVANOV, A., AND SALEH, R. 2003. Design of a switch for network-on-chip applications. *IEEE International Symposium on Circuits and Systems (ISCAS) 5*, 217–220.

PEH, L.-S. AND DALLY, W. J. 1999. Flit-reservation flow control. In *Proceedings of the 6th International Symposium on High-Performance Computer Architecutre (HPCA)*. IEEE Computer Society, 73–84.

PEH, L.-S. AND DALLY, W. J. 2001. A delay model for router microarchitectures. *IEEE Micro 21*, 26–34.

PESTANA, S., RIJPKEMA, E., RADULESCU, A., GOOSSENS, K., AND GANGWAL, O. 2004. Cost-performance trade-offs in networks on chip: a simulation-based approach. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE)*. IEEE, 764–769.

PHILIPS SEMICONDUCTORS. 2002. Device Transaction Level (DTL) Protocol Specification, Version 2.2.

PIGUET, C., JACQUES, HEER, C., O'CONNOR, I., AND SCHLICHTMANN, U. 2004. Extremely low-power logic. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE)*, C. Piguet, Ed. IEEE, 1530–1591.

PIRRETTI, M., LINK, G., BROOKS, R. R., VIJAYKRISHNAN, N., KANDEMIR, M., AND IRWIN, M. 2004. Fault tolerant algorithms for network-on-chip interconnect. In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*. 46–51.

RADULESCU, A., DIELISSEN, J., GOOSSENS, K., RIJPKEMA, E., AND WIELAGE, P. 2004. An efficient on-chip network interface offering guaranteed services, shared-memory abstraction, and flexible network configuration. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE)*. IEEE, 878–883.

RIJPKEMA, E., GOOSSENS, K., AND WIELAGE, P. 2001. A router architecture for networks on silicon. In *Proceeding of the 2nd Workshop on Embedded Systems*. 181–188.

RIJPKEMA, E., GOOSSENS, K. G. W., RADULESCU, A., DIELISSEN, J., MEERBERGEN, J. V., WIELAGE, P., AND WATERLANDER, E. 2003. Trade-offs in the design of a router with both guaranteed and best-effort services for networks-on-chip. In *Proceedings of the Design, Automation and Test in Europe Conference (DATE)*. IEEE, 350–355.

RIXNER, S., DALLY, W. J., KAPASI, U. J., KHAILANY, B., LUPEZ-LAGUNAS, A., MATTSON, P. R., AND OWENS, J. D. 1998. A bandwidth-efficient architecture for media processing. In *Proceedings of the 31st Annual ACM/IEEE International Symposium on Microarchitecture*. 3–13.

ROSTISLAV, D., VISHNYAKOV, V., FRIEDMAN, E., AND GINOSAUR, R. 2005. An asynchronous router for multiple service levels networks on chip. In *Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*. IEEE, 44–53.

SATHE, S., WIKLUND, D., AND LIU, D. 2003. Design of a switching node (router) for on-chip networks. In *Proceedings of the 5th International Conference on ASIC*. IEEE, 75–78.

SIA. 1997. National technology roadmap for semiconductors 1997. Tech. rep., Semiconductor Industry Association.

SIGUENZA-TORTOSA, D., AHONEN, T., AND NURMI, J. 2004. Issues in the development of a practical NoC: The Proteo concept. *Integrat. VLSI J.* Elsevier, 95–105.

SIMUNIC, T. AND BOYD, S. 2002. Managing power consumption in networks-on-chips. In *Proceedings of the Design, Automation and Test in Europe Conference (DATE)*. IEEE Computer Society, 110–116.

SINGH, M. AND NOWICK, S. 2000. High-throughput asynchronous pipelines for fine-grain dynamic datapaths. In *Proceedings of the 6th International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*. IEEE Computer Society, 198–209.

SPARSØ, J. AND FURBER, S. 2001. *Principles of Asynchronous Circuit Design*. Kluwer Academic Publishers, Boston, MA.

STERGIOU, S., ANGIOLINI, F., CARTA, S., RAFFO, L., BERTOZZI, D., AND MICHELI, G. D. 2005. Xpipes lite: A synthesis oriented design library for networks on chips. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE)*. IEEE.

SVENSSON, C. 2001. Optimum voltage swing on on-chip and off-chip interconect. Manuscript available at http://www.ek.isy.liu.se/ christer/ManuscriptSwing.pdf.

SYLVESTER, D. AND KEUTZER, K. 2000. A global wiring paradigm for deep submicron design. *IEEE Trans. Comput. Aided Design Integrat. Circuits Syst. 19*, 242–252.

SYSTEMC. 2002. The SystemC Version 2.0.1. Web Forum (www.systemc.org).

TAMIR, Y. AND FRAZIER, G. L. 1988. High-performance multiqueue buffers for VLSI communication switches. In *Proceedings of the 15th Annual International Symposium on Computer Architecture*. IEEE Computer Society, 343–354.

TAYLOR, M. B., KIM, J., MILLER, J., WENTZLAFF, D., GHODRAT, F., GREENWALD, B., HOFFMAN, H., JOHNSON, P., LEE, J.-W., LEE, W., MA, A., SARAF, A., SENESKI, M., SHNIDMAN, N., STRUMPEN, V., FRANK, M., AMARASINGHE, S., AND AGARWAL, A. 2002. The RAW microprocessor: A computational fabric for software circuits and general-purpose programs. *IEEE MICRO 12*, 2, 25–35.

TORTOSA, D. A. AND NURMI, J. 2004. Packet scheduling in proteo network-on-chip. *Parall. Distrib. Comput. Netw.* IASTED/ACTA Press, 116–121.

VAIDYA, R. S., SIVASUBRAMANIAM, A., AND DAS, C. R. 2001. Impact of virtual channels and adaptive routing on application performance. *IEEE Trans. Parall. Distrib. Syst. 12*, 2 (Feb.) 223–237.

VARATKAR, G. AND MARCULESCU, R. 2002. Traffic analysis for on-chip networks design of multimedia applications. In *Proceedings of the 39th Design Automation Conference (DAC)*. ACM, 795–800.

VSI ALLIANCE. 2000. Virtual component interface standard Version 2. VSI Alliance www.vsi.org.

WANG, H.-S., ZHU, X., PEH, L.-S., AND MALIK, S. 2002. Orion: A power-performance simulator for interconnection networks. In *Proceedings of the 35th Annual ACM/IEEE International Symposium on Microarchitecture*. IEEE Computer Society Press, 294–305.

WEBER, W.-D., CHOU, J., SWARBRICK, I., AND WINGARD, D. 2005. A quality-of-service mechanism for interconnection networks in system-on-chips. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE)*. IEEE, 1232–1237.

WIEFERINK, A., KOGEL, T., LEUPERS, R., ASCHEID, G., MEYR, H., BRAUN, G., AND NOHL, A. 2004. A system level processor/communication co-exploration methodology for multi-processor system-on-chip platforms. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE)*. IEEE Computer Society, 1256–1261.

WIELAGE, P. AND GOOSSENS, K. 2002. Networks on silicon: Blessing or nightmare? In *Proceedings of the Euromicro Symposium on Digital System Design (DSD)*. IEEE, 196–200.

WORM, F., THIRAN, P., MICHELI, G. D., AND IENNE, P. 2005. Self-calibrating networks-on-chip. In *International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2361–2364.

XANTHOPOULOS, T., BAILEY, D., GANGWAR, A., GOWAN, M., JAIN, A., AND PREWITT, B. 2001. The design and analysis of the clock distribution network for a 1.2 GHz alpha microprocessor. In *Digest of Technical Papers, IEEE International Solid-State Circuits Conference, ISSCC*. IEEE, 402–403.

XU, J., WOLF, W., HENKEL, J., AND CHAKRADHAR, S. 2005. A methodology for design, modeling, and analysis of networks-on-chip. In *International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1778–1781.

XU, J., WOLF, W., HENKEL, J., CHAKRADHAR, S., AND LV, T. 2004. A case study in networks-on-chip design for embedded video. In *Proceedings of Design, Automation and Testing in Europe Conference (DATE)*. IEEE, 770–775.

ZHANG, H., GEORGE, V., AND RABAEY, J. M. 1999. Low-swing on chip signaling techniques: Effectiveness and robustness. *IEEE Trans. VLSI Syst. 8*, 3 (Aug.) 264–272.

ZHANG, H., PRABHU, V., GEORGE, V., WAN, M., BENES, M., ABNOUS, A., AND RABAEY, J. M. 2000. A 1 V heterogeneous reconfigurable processor IC for baseband wireless applications. In *International Solid-State Circuits Conference. Digest of Technical Papers (ISSCC)*. IEEE, 68–69.

ZIMMER, H. AND JANTSCH, A. 2003. A fault tolerant notation and error-control scheme for switch-to-switch busses in a network-on-chip. In *Proceedings of Conference on Hardware/Software Codesign and System Synthesis Conference CODES ISSS*. ACM, 188–193.

Received September 2004; revised September 2005; accepted January 2006